APPENDIX A

POTHOLE-RIVER NETWORKED WATERSHED MODEL (PRINET)

A-1. Technical Model Description

A-2. PRINET Source Code

APPENDIX A-1

Technical Model Description

A. POTHOLE-RIVER NETWORKED WATERSHED MODEL

This appendix contains an overview of the Pothole-River Networked Watershed Model (PRINET), the model that was used to simulate runoff in the Devils Lake basin. The chapter details the geometric inputs to the model, and how those inputs were modified in certain cases. Then the structure of the model tables is explained, as is the generation of restoration scenarios.

Other inputs to the model and their structure are also explained. These inputs include future climate sequences, precipitation and snowmelt data, evaporation data, and the use of lakes in the model. The model outputs and their structure are also covered in detail. Finally, a flowchart together with some notes is presented. These show how the model performs its computations.

A.1. MODEL OVERVIEW

Pothole-River Networked Watershed Model (PRINET) is a hydrologic model that utilizes topographic and climatic information to simulate a long-term process (generally from 20 to 50 years) of rainfall, evaporation, and water storage for a terrain with a substantial number of depressions (or potholes). It was designed to model in the Devils Lake watershed the impact of depressions on runoff. PRINET could conceivably be applied to another watershed if the topographic and climatic information were available.

The PRINET application was written in Microsoft Visual Basic 6.0 (Visual Basic For Applications), inside a Microsoft Access database. Microsoft Access 2000 is required to run the application.

There are two basic types of simulations that were computed: (1) calibration simulations (labeled COMP), and (2) future simulation climate sequences (labeled 001 through 010, and WET). The documentation in this chapter applies to both types of simulations.

A.1.1. Calibration Simulations

Calibration simulations were used to set each subwatershed's parameters (such as infiltration rate and the evapotranspiration multiplier) to attempt to match the physical parameters at geographic locations. These physical parameters were deduced on the basis of values recommended for use by other studies, texts, and reports, and by PRINET's predicted flows when compared to observed historical streamflows.

A.1.2. Future Simulations

Once the physical parameters were calibrated, these parameters were used to predict future runoffs in the Devils Lake watershed. A set of climatic future conditions was input and PRINET was used to predict future runoff values. Furthermore, because PRINET uses geographic inputs, it was able to predict the effects of changes in these geographic inputs on future runoff values.

A.2. INPUT DATA

The inputs to PRINET are the attribute tables (database tables which provide geographic information) of themes developed in ArcView and HEC-GeoHMS (HEC-GeoHMS User's Manual, July 2000). The naming conventions for the tables are as follows:

- <SUBWATERSHED> is replaced by the name of the subwatershed being modeled, for example EDMORE or HURRLAKE
- <SEQ> is replaced by the climate sequence: 001, 002, 003, 004, 005, 006, 007, 008, 009, 010, or WET.
- <RESTSCENARIO> is replaced by the letter that indicates the restoration scenario: A, B, C, D or E.

For example,

OUTPUT_<SUBWATERSHED>_<SEQ>_<RESTSCENARIO>

Would represent tables such as

OUTPUT_EDMORE_003_A

or

OUTPUT_STARKWEATHER_WET_E.

In the following sections, a table name is provided as the section heading. The table's field names and their descriptions are list under each table name. Some tables in the database may have fields that are not listed. These possible additional fields were not used to develop the PRINET models. The ArcView themes from which these tables were extracted have the same name as the tables in the database, but without the IMP_ prefix.

Models were created by subwatershed (six subwatersheds were modeled). A total of five tables imported from ArcView into Microsoft Access were needed to create each subwatershed model. In addition, one other table was required for each restoration scenario for each subwatershed; however, this table was not imported from ArcView but was created by PRINET.

The watershed delineation routines in HEC-GeoHMS were used to develop a "River" theme and a "Watershed" theme—these are the names HEC-GeoHMS assigns to them; these themes were renamed to uniquely identify each subwatershed modeled.

IMP_<SUBWATERSHED>_WSHEDS

This is the attribute table from the "Watershed" theme created by HEC-GeoHMS. Each table corresponds to one subwatershed model. Six different subwatersheds were delineated: Comstock, Hurricane Lake, Mauvais 6100, Mauvais, Edmore, and Starkweather.

Each watershed is divided into numerous subbasins. The watersheds were delineated to provide an average subbasin area of 0.3 square miles. The following figure illustrates the IMP_COMSTOCK_WSHEDS subwatershed theme (the smallest of the subwatersheds modeled, COMSTOCK has 257 subbasins):

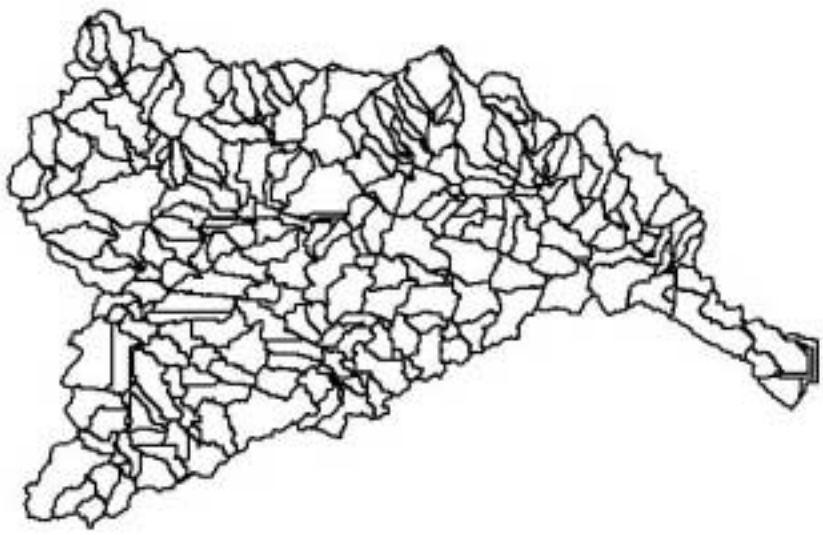

Figure A-1. Comstock subwatershed divided into individual subbasins by HEC-GeoHMS.

The attribute table's fields required by PRINET are:

- WSHID (Number): The subbasin ID for the subbasin, as determined by HEC-GeoHMS.
- AREA_M2 (Number): The area in square meters of the subbasin.
- Region (Number): The region number, which also appears in the Regimes table for the subwatershed model. PRINET allows the user to set different parameters for the different subbasins. These regions are set in ArcView on the basis of geographic location, and the IMP_<SUBWATERSHED>_WSHEDS table contains these region numbers for use in the program. Region numbering must begin at 1 and be consecutive. For example, a model with only one region would have all subbasins set at Region = 1. A model with three regions would use Region numbers 1, 2 and 3.
- PrecipGage (Text): The name of the precipitation gage used for this subbasin. The table PrecipGages provides the relationship between the precipitation gage names and the tables in which the precipitation gage information is stored.

IMP_<SUBWATERSHED>_RIVER

This is the attribute table from the RIVER theme created by HEC-GeoHMS. The river theme is a network theme of river segments, as illustrated in the following figure.

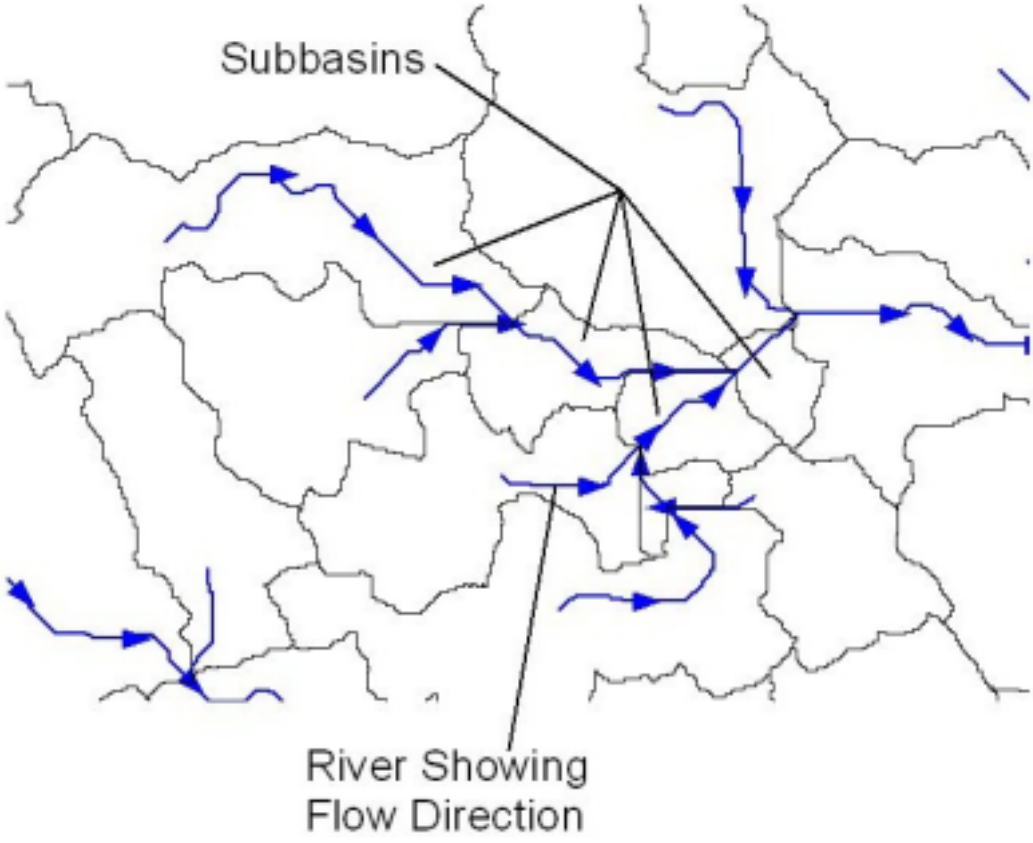

Figure A-2. Networked river segments developed by HEC-GeoHMS delineation

Each river segment corresponds to exactly one subbasin. The RIVER table contains the information regarding the ordering of the rivers (and therefore the ordering of the subbasins). Each field in the RIVER table gives the attributes from one river segment.

The RIVER table has numerous fields; however, only the following are used by PRINET:

- FROM_NODE (Number): The upstream node number.
- TO_NODE (Number): The downstream node number.
- WSHID (Number): The subbasin ID from HEC-GeoHMS in which the river segment is contained. Each subbasin has exactly one river segment assigned to it.
- RIVID (Number): The river ID for the river segment, it is the same as the WSHID value; that is each river segment has the same ID number as the subwatershed it is contained in.

IMP_<SUBWATERSHED>_DEPTABLE_<RESTSCENARIO>

This table contains a list of all the depressions in the subwatershed and their characteristics. It contains the following fields:

- POLYID (Number): An integer that is unique to each depression.
- DEPTH_M (Number): The average depth in meters of the depression.
- AREA_M2 (Number): The surface area of the depression in square meters.
- DEPTYPE123 (Number): The depression classification: 1 if intact, 2 if drained.
- VOLM3 (Number): The total volume in cubic meters of the depression.
- INTERRIV (Boolean): True (= -1) if the depressions intersects the river network delineated by HEC-GeoHMS, False (= 0) if the depression does not intersect the river network delineated by HEC-GeoHMS. The following figure illustrates how river intersection was determined:

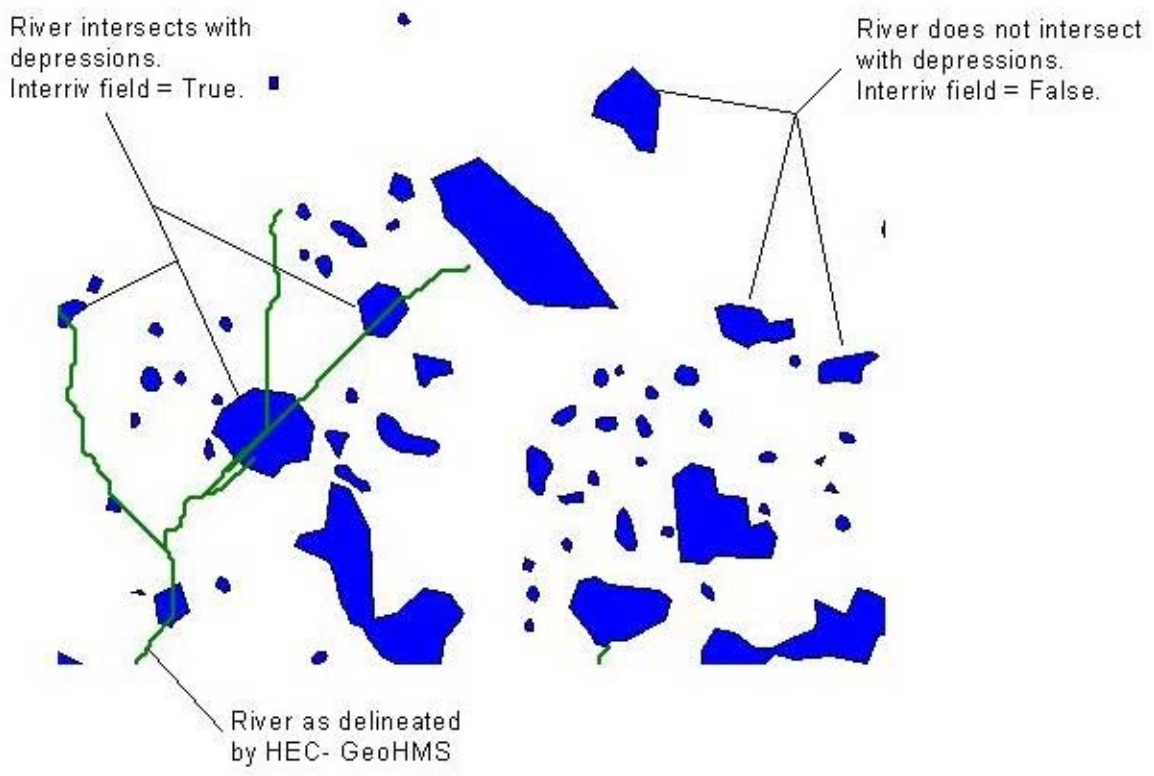

Figure A-3. Representative area showing depressions and their intersection with the river theme delineated by HEC-GeoHMS.

Following is an example of one record of a typical DEPTABLE, as it appears after it is imported from ArcView:

POLYID	DEPTH_M	AREA_M2	DEPTYPE123	VOLM3	INTERRIV
56676	0.2742	3012.7	1	826.0823	FALSE

The restoration scenario tables, which will be described later, are named similarly to this table, but have a suffix of "B," "C," "D," or "E" instead of "A." These tables are not imported from ArcView but are created by PRINET. Furthermore, during PRINET's creation of these additional tables, 3 additional fields are added to the IMP_<SUBWATERSHED>-_DEPTABLE_A table. These three additional fields will be described later, in Section A.2.3, "Creation of Restoration Scenarios from a Model".

IMP_<SUBWATERSHED>_PPOINTLOCATIONS

This table is the attribute table of an intersection between the pour point location of each depression and the subwatershed theme. It provides the location of each depression's pour point by subbasin. It contains the following fields:

- POLYID (Number): An integer, which is unique to each depression.
- WSHID (Number): The subbasin ID of the subbasin delineated by HEC-GeoHMS.

The pour points themselves were generated by the following process:

(1) The small stream grid (as generated by HEC-GeoHMS) for the subwatershed model was multiplied by the flow accumulation grid. This provided a grid that looked like the small stream grid, but whose grid values were the flow accumulation values.

(2) This grid was then converted to a polygon theme, and the polygon theme was converted to a point theme, using the centers of each polygon. The result was a point theme with flow accumulation values as the attributes.

(3) The point theme was then intersected with the depression theme, which provided a theme with an attribute table showing the location of each point by the depression POLYID. For each depression, the point inside the depression with the maximum flow accumulation was selected, and called the pour point of the polygon. However, many polygons, primarily the small ones that were off the river network, did not intersect the small stream grid. In these cases, the centers of the depression polygons were taken to be the pour points.

The following figure illustrates pour points for some sample depressions:

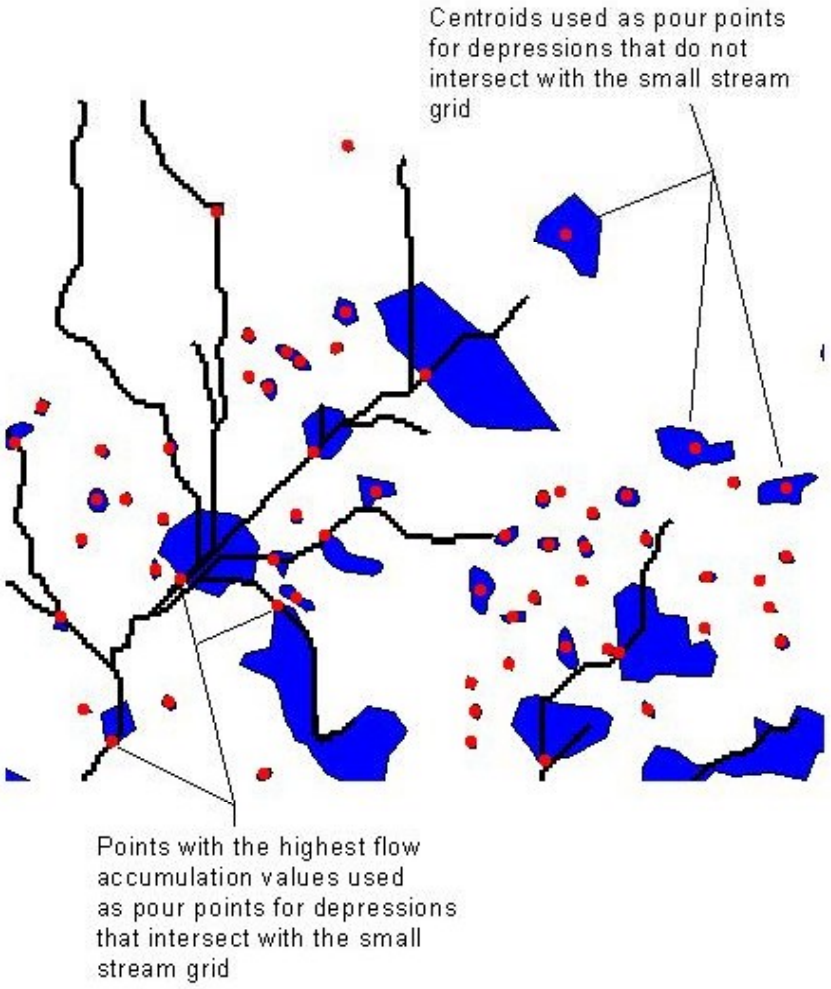

Figure A-4. Representative area showing depressions and their pour points as defined based on intersection with the small stream grid.

IMP_<SUBWATERSHED>_PPOINTWSHEDS

This is the attribute table from a theme that contains the contributing areas for each depression, excluding the depressional surface area.

The ArcView theme from which this attribute table was extracted was developed by a multi-step process. To the set of pour points (one point for each intact and drained depression— the generation of these pour points was explained previously), the outlet point of the subwatershed was added. Then, subbasins were delineated for each of the points. A script obtained from the ESRI website (http://support.esri.com/) was used to accomplish this delineation (the script is called wshed_point.ave, dated 25/Feb/2000, updated: 20/Apr/2000, by Fridjof Schmidt).

The result was a complete coverage of the subwatershed, with small subbasins delineated on each of the pour points. This provided a contributing drainage area for each of the depressions. The following figure, a detail from the Comstock subwatershed, illustrates the result.

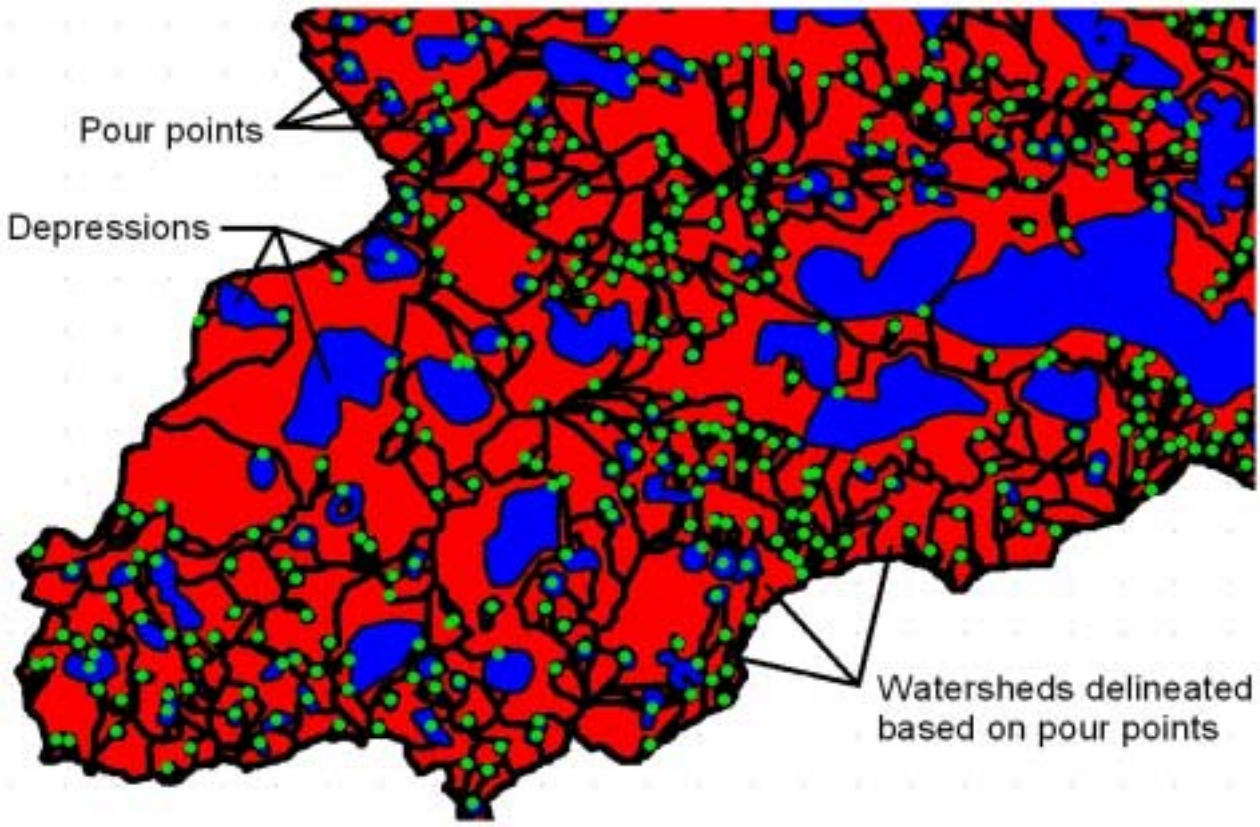

Figure A-5. Detail of Comstock subwatershed showing delineation of individual subbasins for each depression pour point (in red).

Then, from this theme of small subbasins, the area where the depressions (both intact and drained) themselves were located was cut out (removed) from the theme. The depressional area was removed in order to provide contributing areas that were not overlapping the depressions themselves.

The resulting theme resembles the overall subwatershed area, but with holes where the depressions were located. Since the contributing areas for each depression may have been broken up, multiple areas may appear for the same POLYID (the depression identifier) in the attribute table. The following figure illustrates this theme for the Comstock subwatershed.

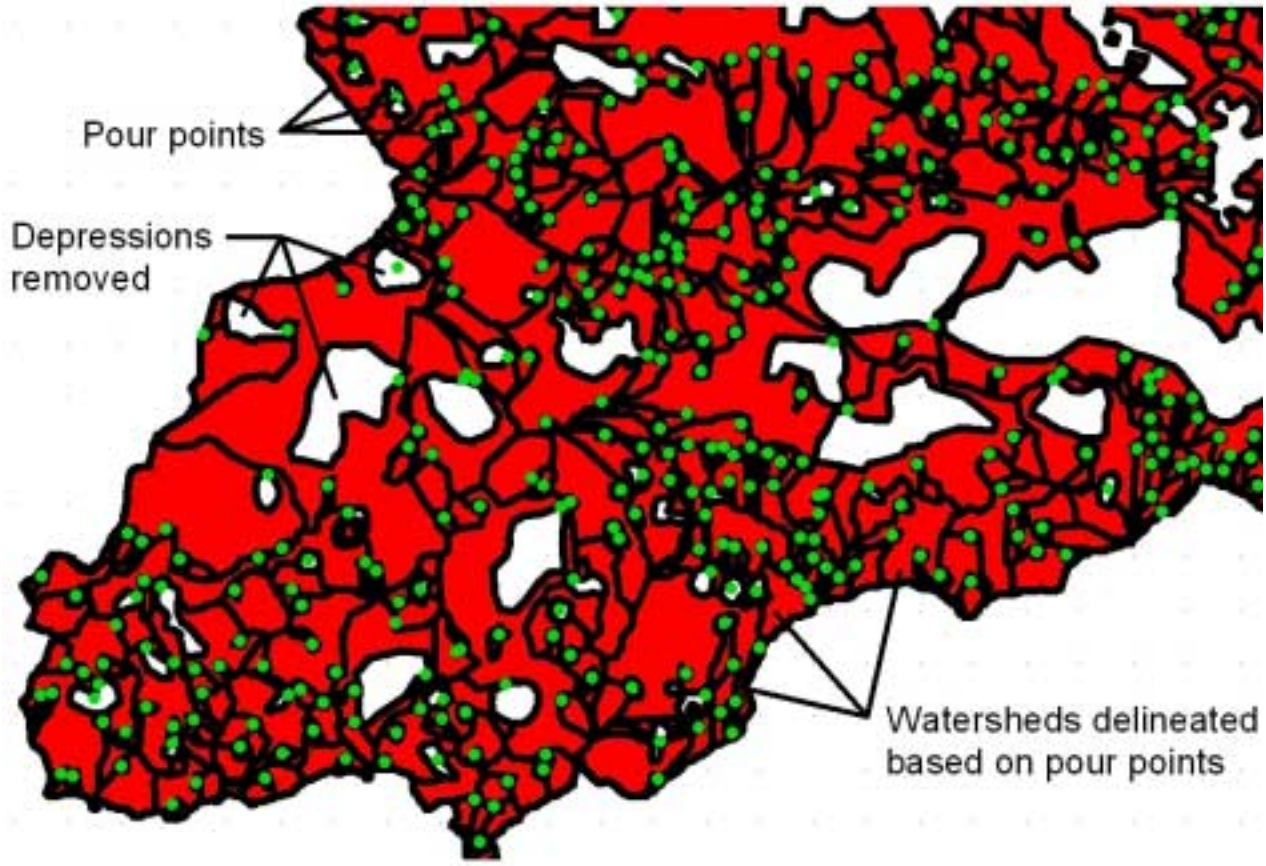

Figure A-6. Detail of Comstock subwatershed showing delineation of individual subbasins for each depression pour point, with depressions removed.

Therefore, the sum of all areas for each POLYID provides the contributing area for each depression. The fields required by PRINET are:

- POLYID (Number): An integer, which is unique to each depression.
- AREA_M2 (Number): The surface area of the depression in square meters when the depressions are full.

A.2.1. Adjustments to the input geometric data

For the six subwatersheds modeled, once the five tables were imported into the Microsoft Access database, some tables were modified before the subwatershed model was generated. There were two adjustments to the imported tables that were performed on some of the subwatersheds:

A.2.1.1. Changing Location of Depressions

Each depression is assigned to a subbasin based on the location of its pour point. It may be clear by examination of the ArcView coverage of the depressions and the subbasins, that the location

of the pour point of a depression does not best represent the subbasin to which that depression should be assigned.

If this is the case, then in the IMP_<SUBWATERSHED>_PPOINTLOCATIONS table, the assignments of depressions can be changed from one subbasin to another. As a generic example, a few records of this table might look like this:

POLYID	WSHID
22775	269
22784	242
22785	232
22786	256

Based on the geometry of the depression 22775, it might be logical to shift it from its current assignment in Wshid 269, to another Wshid, say 270. The Wshid field would simply be modified to the new Wshid number:

POLYID	WSHID
22775	270
22784	242
22785	232
22786	256

The Wshid number corresponds to the numbers that HEC-GeoHMS assigns to each of the subbasins it creates. The new Wshid number assigned to any polygon should be an actual subbasin listed in the IMP_<SUBWATERSHED>_WSHEDS table.

A.2.1.2. Re-sequencing rivers

It may be apparent from the aerial photographs that HEC-GeoHMS did not correctly sequence the rivers. For example, in the Edmore subbasin, the following was the original sequencing for River segments 1281 and 1304:

ARCID	GRID_CODE	FROM_NODE	TO_NODE	WSHID	RIVID	LENGTH	RIV_LENGTH
1278	1281	1277	1297	1281	1281	670	669.8
1294	1304	1306	1297	1304	1304	382	381.6

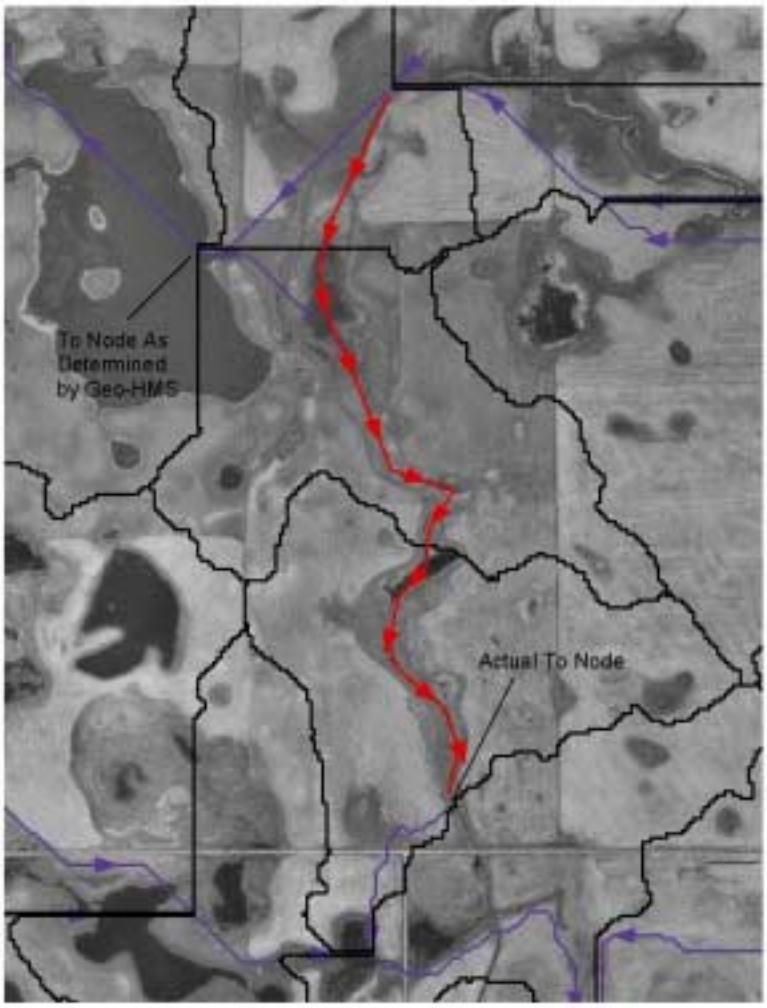

Figure A-7. Example of river resequencing in Edmore subbasin

But from the aerial photographs, it can be seen the HEC-GeoHMS has not correctly sequenced the rivers (this is certainly due to the Digital Elevation Model, not to any flaw in HEC-GeoHMS). The rivers should flow south, not west. So their "TO_NODE" values were changed as follows:

ARCID	GRID_CODE	FROM_NODE	TO_NODE	WSHID	RIVID	LENGTH	RIV_LENGTH
1278	1281	1277	1369	1281	1281	670	669.8
1294	1304	1306	1369	1304	1304	382	381.6

A.2.2. Creation of a Model From Input Tables

Using the five input tables described in the previous section, the "Create New Model From ArcView Tables" button on the program's main screen (the RunSimul form) creates a set of tables for the subwatershed model: a subwatershed table, a regime table and a river table.

The tables generated and the naming conventions are described below, where <SUBWATERSHED> is replaced by the name of the subwatershed being modeled. The fields and their definitions are listed in each table name. Any fields found in the tables that are not on the list are not required by PRINET.

MODEL_<SUBWATERSHED>_<RESTSCENARIO>_WSHEDS

Five fields from this table can be modified by the user after PRINET creates the table: InitWaterM3, InputStreamTable, LakeID, ExtraInflowID, and ExtraOutflowID. All other fields should be left with their original values.

- SubBSeq (Number): The subbasin sequence number. This is the subwatershed model's unique identifier for each subbasin. The subbasins are numbered so that subbasins always have lower subbasin sequence numbers than the downstream subbasins. The sequencing information for the subbasins is extracted from the imported river table (IMP_<SUBWATERSHED>_RIVER), which has the *from* and *to* nodes for each subbasin.

- WShid (Number): The subbasin ID from HEC-GeoHMS. This WShid identifier is used during the creation of the WSHED table from the imported ArcView tables (prefixed IMP), but once the subwatershed model's WSHED table is generated, PRINET does not use the WShid field. It can be used by the user to provide a relationship between the HEC-GeoHMS WShid and the SubBSeq numbers.

- DSSubBSeq (Number): The subbasin sequence number of the subbasin downstream of the SubBSeq subbasin.

- PrecipGage (Text): The name of the precipitation gage used for this subbasin. The table PrecipGages provides the relationship between the precipitation gage names and the tables in which the precipitation gage information is stored.

- Region (Number): The region number, which also appears in the Regimes table for the subwatershed model. PRINET allows the user to set different parameters for the different subbasins. These regions are set in ArcView on the basis of geographic location, and the IMP_<SUBWATERSHED>_WSHEDS table contains these region numbers for use in the program. Region numbering must begin at 1 and be consecutive. For example, a model with only one region would have all subbasins set at Region = 1. A model with three regions would use Region numbers 1, 2 and 3.

- InitWaterM3 (Number): The initial amount of water in cubic meters in the depressions at the start of the simulation. The program proportions this water between the on-river and off-river depressions at the start of the simulation. The amount proportioned is based on the relative volume of the on-river and off-river depressions. If the on-river depressions had 75% of the depressional volume then the program would start with them containing 75% of the initial water specified in this field; the balance would go to the off-river depressions. The default value is 0: different values can be added by the user.

- InputStreamTable (Text): A table that provides source flows to the subwatershed at specific subbasins. Generally, this will be the output from another model. Only the subwatershed model name from which the flows will be obtained needs to be specified. For example, if "HURRLAKE" is specified in this field, then the simulation will look for a table called OUTPUT_HURRLAKE_COMP_A, in the case of the calibration run (COMP) and restoration scenario A. These input tables require two fields: Date, which is the calendar day, and CMPD, which is the cubic meters of flow per day. All flows from this input table will be added to the subbasin record where the InputStreamTable is specified. The default for this field is null; any value for this field is manually added by the user.

- LakeID (Number): The ID number of any lake that exists at this subbasin. The existence of a lake at a subbasin causes the program to use the storage-outflow relationships specified in the Lakes table instead of using the normal algorithm to calculate the outflow of the on-river depressions. The default for this field is null; any value for this field is manually added by the user.

- ExtraInflowID (Text): The name of a table to which PRINET should write the inflows to a subbasin. The table name will be prefixed by "INFLOW" and suffixed by the simulation run and restoration scenario. For example, if the word "GAGENAME" was entered in this field for a particular subbasin, the table INFLOW_GAGENAME_<SEQ>_<RESTSCENARIO> would be created during the simulation and it would contain the flows into the subbasin on a daily basis. This table has two fields, Date and CMPD (the flow in cubic meters). A maximum of 25 inflow tables may be specified. The default for this field is null; any value for this field is manually added by the user.

- ExtraOutflowID (Text): This is similar to the previous field, except it contains the name of a table to which PRINET should write the outflows from the subbasin. The table name will be prefixed by OUTFLOW and suffixed by the simulation run and restoration scenario. For example, if the word "GAGENAME" was entered in this field for a particular subbasin, the table OUTFLOW_GAGENAME_COMP_A would be created during the simulation, it would contain the flows into the subbasin on a daily basis. This table has two fields, Date and CMPD (the flow in cubic meters). A maximum of 25 outflow tables may be specified. The default for this field is null; any value for this field is manually added by the user.

- WSAreaM2NonDep (Number): The area of the subbasin, not including intact depressional surface area, in square meters. This is calculated by adding all the contributing areas of the depressions whose pour points are located in the subbasin. These contributing areas do not include the area of intact depressions but do include the surface area of drained depressions.

- DepVolM3 (Number): The total storage volume of all intact depressions in the subbasin, in cubic meters. A depression is defined as being in a subbasin if its pour point is inside the subbasin.

- DepAreaM2 (Number): The total surface area of intact depressions in the subbasin, in square meters.

- WSAreaNonDepOnRiverM2 (Number): This is analogous to WSAreaM2NonDep, but it only has the contributing areas for those depressions (both intact and drained) that intersect the HEC-GeoHMS river network.

- DepVolOnRiverM3 (Number): This is analogous to DepVolM3, but it only includes the volumes of those intact depressions that intersect the river network.

- DepAreaOnRiverM2 (Number): This is analogous to DepAreaM2, but it only includes the surface area of those intact depressions that intersect the river network.

- WSAreaNonDepOffRiverM2 (Number): Analogous to WSAreaNonDepOnRiverM2, but this is for depressions that do not intersect the river network.

- DepVolOffRiverM3 (Number): Analogous to DepVolOnRiverM2, but this is for depressions that do not intersect the river network.

- DepAreaOffRiverM2 (Number): Analogous to DepAreaOnRiverM2, but this is for depressions that do not intersect the river network.

- NumDeps (Number): The total number of depressions assigned to this subbasin.

- NumDepsOnRiver (Number): The total number of depressions assigned to this subbasin that intersect the river network.

- NumDepsOffRiver (Number): The total number of depressions assigned to this subbasin that do not intersect the river network.

The following shows one record from this table from the MAUVAIS model. For this record, Irvine Lake was assigned to the subbasin, (LakeID = 6 is for Irvine Lake), and inflow and outflow tables were specified in the ExtraInflowID and ExtraOutflowID fields. This will cause tables to be created during the simulation runs that give the daily inflows and outflows of this subbasin. The rows shown all belong to the same record.

SubBSeq	WShid	DSSubBSeq	PrecipGage	Region	InitWaterM3	InputStreamTable
2888	2754	2898	CHURCHF	4	0	

LakeID	ExtraInflowID	ExtraOutflowID	WSAreaM2NonDep	DepVolM3	DepAreaM2
6	IRVINE	IRVINE	349141.2	16566610	14282790

WSAreaNonDepOnRiverM2	DepVolOnRiverM3	DepAreaOnRiverM2
349141.2	16566610	14282790

WSAreaNonDepOffRiverM2	DepVolOffRiverM3	DepAreaOffRiverM2
0	0	0

NumDeps	NumDepsOnRiver	NumDepsOffRiver
1	1	0

MODEL_<SUBWATERSHED>_REGIME

The fields from this table can be modified by the user to reflect the subwatershed model's desired parameters, with the following restrictions: (1) there must be exactly as many Regions specified in the table as appear in the IMP_<SUBWATERSHED>_<RESTSCENARIO>-_WSHEDS table's Region field; (2) the region numbering must begin with 1 and be consecutive up to the number of regions present, and; (3) there must be exactly two RegimeID's for each Regime, with values of 1 and 2.

- Region (Number): This is the geographic region number, which corresponds to the Regions assigned to each subbasin in IMP_<SUBWATERSHED>_WSHEDS table, and in the MODEL_<SUBWATERSHED>_<RESTSCENARIO> _WSHEDS table.

- RegimeID (Number): There should be two RegimeID records for each region. The first record should have RegimeID = 1 and corresponds to the low infiltration/winter regime. The second record should have RegimeID = 2 and corresponds to the high infiltration/summer regime.

- Name (Text): The name of the regime, not used by the program but for the convenience of the user. "Low" is used for RegimeID = 1 and "High" for RegimeID = 2.

- EntryThresholdF (Number): This is the threshold temperature which determines the transition from one regime to the other. Each precipitation gage, for each day of record, is assigned to either Regime 1 (cold season) or Regime 2 (warm season). Subbasins assigned to these gages use the corresponding assignments of their precipitation gage to determine the set of parameters (i.e. maximum infiltration rate, percolation rate) used for a particular day. Seasons (i.e., high and low infiltration regimes) are calculated for each Precipitation Gage record used in the subwatershed model. These gages have a temperature record. When the 30-day moving average of the average daily temperature (the average of the high and low temperatures recorded) crosses the EntryThresholdF

value, then the Regime is switched on that date for that gage. For example, suppose the entry threshold for Regime 1 is set at 35 degrees F. If for a particular gage, on 11/5/1993, the moving average of the temperature dropped below 35 degrees F for the first time that winter, then 11/5/1993 would be classified as Regime 1, while the previous day, 11/4/1993, would be in Regime 2. All subbasins assigned to that gage would use the regimes that were calculated for that gage. As an example of transitioning into Regime 2, suppose that the EntryThresholdF for Regime 2 was set at 45. When the 30-day moving average of temperature for a gage first exceeded 45 degrees, then from that day forward the Regime would be set at 2 for that particular gage. However, there is smoothing function which is used in the final days of each low infiltration Regime (Regime = 1) to smooth the transition to the higher infiltration (Regime = 2) regime. This smoothing function affects the infiltration only, and prevents it from changing suddenly from one day to the next. The infiltration begins to rise from the Regime = 1 value toward the Regime =2 value once the 30-day moving average temperature goes above 32 degrees.

- InfilMPD (Number): The maximum potential infiltration in meters per day. During the simulation, actual potential infiltration is reduced with increasing saturation of the soil.

- PercMPD (Number): The potential percolation in meters per day. Percolation only takes place if the soil capacity is less than upper zone depth (another parameter set by Regime, defined later). The soil capacity is defined as the soil "dryness" depth, it is the amount of water the soil can still absorb before becoming completely saturated. It changes daily and equals the potential soil capacity minus the soil moisture. So percolation only takes place when the soil is saturated enough.

- ETCoef (Number): The dimensionless evapotranspiration coefficient. The actual evapotranspiration rate is reduced with increasing soil capacity (dryness). The ET coefficient should be between 0 and 1.

- UpperZoneM (Number): A depth, in meters, where if the soil capacity is less than this amount, then percolation is allowed to take place. Percolation only takes place when the soil capacity is low, that is, the soil saturation is high.

- PotentialSoilCapM (Number): The total capacity of the soil to hold water, in meters of water.

- InitialSoilCapM (Number): The initial soil capacity, in meters, to apply at the start of each simulation.

- NextDayFractionalFlow (Number): A dimensionless attenuation factor which should be less than 1 and greater than or equal to zero. PRINET calculates total potential outflow from a subbasin for a particular day. A fraction of the flow indicated in this field is lagged and appears the next day at the downstream subbasin. The rest of the outflow fraction (1 – NextDayFractionalFlow), appears the same day at the downstream subbasin. This provides each subbasin with a 2-step hydrograph, which can be used to attenuate flows to match the shapes of observed streamflows.

A typical table would appear as follows for a model with 3 Regimes (the table is broken in two due to space limitations, each of the rows of the 2nd part correspond to the rows in the first part):

Region	RegimeID	Name	EntryThresholdF	InfilMPD	PercMPD	ETCoef	UpperZoneM
1	1	Low	35	0.02	0	0.76	0.01
2	1	Low	35	0.008	0	0.76	0.01
3	1	Low	35	0.008	0	0.76	0.01
1	2	High	45	0.36	0.005	0.95	0.03
2	2	High	45	0.16	0.005	0.83	0.03
3	2	High	45	0.34	0.005	0.95	0.03

PotentialSoilCapM	InitialSoilCapM	NextDayFractionalFlow
0.33	0.23	0.1
0.33	0.23	0.1
0.33	0.23	0.1
0.33	0.23	0.1
0.33	0.23	0.1
0.33	0.23	0.1

MODEL_<SUBWATERSHED>_RIVER

This table is used during the generation of the subwatershed model by PRINET. Several fields are added by PRINET to the imported river table (IMP_<SUBWATERSHED>_RIVER) to create the subwatershed model river table (MODEL_<SUBWATERSHED>_RIVER). PRINET uses these added fields to store values which are then transferred into the (MODEL_-<SUBWATERSHED>_<RESTSCENARIO>_WSHED) table. Once the subwatershed model is created, PRINET does not use the MODEL_<SUBWATERSHED>_RIVER table but it remains available should the user care to examine it.

In addition to all the fields contained in the subwatershed model's RIVER table (described previously under the heading IMP_<SUBWATERSHED>_RIVER), the following fields are added:

- Layer (Number): For each subbasin, this is the sequential number for subbasins upstream of the subwatershed's outlet. The subbasin closest to the outlet has a layer number of 1, those immediately upstream have a layer number of 2. Those subbasins most upstream will have the highest number. This layer sequencing is used only to develop the SubBSeq and DSSubSeq numbers, which are described below.

- SubBSeq (Number): Each subbasin in a subwatershed is assigned this unique identifier. The SubBSeq numbers are assigned by sorting the subbasins first by reverse layer number, so that those subbasins that are the most upstream are assigned the lowest SubBSeq number. This numbering guarantees that a subbasin's SubBSeq number cannot

be higher than any downstream subbasin's SubBSeq. The subbasins most upstream have the lowest SubBSeq number.

- DSSubSeq (Number): For each subbasin, this corresponds to the SubBSeq number of the downstream subbasin. The most downstream subbasin has a DSSubBSeq number of 0, which represents the outlet of the subwatershed.

- DistFromOutM (Number): This field is not used.

A.2.3. Creation of Restoration Scenarios from a Model

A restoration scenario is one in which a quantity of depressions which had been classified as drained (which are assumed not to hold water) are converted to intact depressions (which are assumed to hold water). These restoration scenarios are used in the future simulations to estimate the impact of restoring drained depressions on future runoff.

Once a model was created from the input tables, the restoration scenarios for this model were developed. This was accomplished by clicking the "Create All Restoration Scenarios" button on the PRINET form. Note that because PRINET uses random numbers in the generation of these restoration scenarios, that clicking "Create All Restoration Scenarios" will not create the same restoration models when it is clicked a second time for the same subwatershed. This is described in detail below.

Restoration scenarios involve the conversion of some drained (Deptype123 = 2) depressions into intact (Deptype123 = 1) depressions. These are used so that re-running PRINET on the subwatershed can indicate how the restoration scenarios affect the runoff.

Restoration scenarios were developed by assuming that all depressions greater than or equal to 0.5 feet (0.1524 meters) of average depth constitute the group of restoration candidates. The total depressional volume of this group is the total restoration volume.

All simulations started on 10/1/1978, with the original calibration geometry. When the simulation date reached the restoration transition date (the date at which restoration of drained depressions is assumed to be implemented) which was 10/1/2003, PRINET reads the parameters from the MODEL table corresponding to the restoration being simulated (if not scenario A). These parameters are used for all simulations from 10/1/2003 forward.

Each depression belonging to the group of restoration candidates was assigned a random fractional number between 0 to 1 by PRINET. The depressions were sorted according to these random numbers. Using this sort order, PRINET calculated the cumulative volume as a fraction of the group total, and the cumulative area as a fraction of group total.

For example, the first few records of a group of restoration candidate might appear as follows (this example is from IMP_EDMORE_DEPTABLE_A):

POLYID	DEPTYPE123	DEPTH_M	AREA_M2	VOLM3	INTERRIV	RandomNum	RunningTotalFracVol	RunningTotalFracArea
18841	2	0.5476	10247	5611.2572	FALSE	0.000077	0.000113268	0.000103462
27187	2	0.407	62341.5	25372.991	TRUE	0.000103	0.000625444	0.000732914
39687	2	0.1752	50505.6	8848.5811	TRUE	0.000303	0.000804061	0.001242861
43664	2	0.4933	103317.3	50966.424	TRUE	0.002125	0.001832863	0.002286039
38532	2	0.2466	116337.9	28688.926	TRUE	0.002598	0.002411974	0.003460684
38847	2	0.4382	40467.9	17733.034	FALSE	0.002713	0.002769931	0.003869282
32410	2	0.3466	9991.2	3462.9499	FALSE	0.003188	0.002839834	0.003970161

The RandomNum field has the random numbers calculated by PRINET which have been assigned to each depression which is a restoration candidate. In the table above, they are sorted by this random number. The field RunningTotalFracVol gives the portion of the volume so far, as a fraction of the total volume of all restoration candidates.

For this subwatershed (EDMORE), the total volume of the restoration candidate group (the total of VOLM3 of all drained depressions, i.e., DepType123 = 2, with depth greater than or equal to 0.1524 meters) is 49,539,577.832 cubic meters (this value is not shown in the previous table). The first restoration candidate has PolyID = 18841. It has a volume of 5611.2572 cubic meters, so the RunningTotalFracVol = 5611.2572 / 49,539,577.832 = 0.000113268. The 2nd restoration candidate has PolyID = 27187. It has a volume of 25372.991 cubic meters, so the RunningTotalFracVol = (All previous depressional volume + current depression's volume)/(Total restoration volume) = (5611.2572 + 25372.991)/(49,539,577.832) = 0.000625444.

Similarly, the RunningTotalFracArea field provides a running total of the fractional area of the depressions, as a fraction of the total area of the restoration candidate group.

Four levels of restoration are simulated for each model, as described in the following table:

Restoration Scenario	Description
A	The original model, as calibrated.
B	25% of the total restoration volume is converted from drained to intact.
C	50 % of the total restoration volume is converted from drained to intact.
D	75% of the total restoration volume is converted from drained to intact.
E	100% of the total restoration volume is converted from drained to intact (i.e., all restoration candidates are converted from drained to intact).

The restoration scenarios are assigned by the running total fractional volumes calculated previously. If the running total fractional volume is less than or equal to the fraction of restoration for each restoration scenario, then those depressions are "restored" (changed from drained to intact) for that restoration scenario. For example, for restoration scenario C, all depressions whose running total fractional volume is less than or equal to 0.5 will be changed from drained to intact for the C model. In the following table (from IMP_EDMORE_DEPTABLE_A), PolyIDs 39999 and 45013 are changed to intact in the C model because their running total fractional area is less than or equal to 0.5, but 48402 and 39988 are not changed.

POLYID	DEPTYPE123	DEPTH_M	AREA_M2	VOLM3	INTERRIV	RandomNum	RunningTotalFracVol	RunningTotalFracArea
39999	2	0.168	2405.7	404.1576	FALSE	0.442525208	0.499649256	0.485442458
45013	2	0.225	1174	264.15	FALSE	0.442694128	0.499654588	0.485454311
48402	2	0.4151	59570.2	24727.59	FALSE	0.442749977	0.500153736	0.486055782
39988	2	0.3315	134254.5	44505.367	TRUE	0.443671823	0.501052116	0.487411328

The following shows a few records from IMP_EDMORE_DEPTABLE_C for the corresponding records shown above, note how the 39999 and 45013 depressions were changed to DepType123 = 1 (intact):

POLYID	DEPTH_M	AREA_M2	DEPTYPE123	VOLM3	INTERRIV
39999	0.168	2405.7	1	404.1576	FALSE
45013	0.225	1174	1	264.15	FALSE
48402	0.4151	59570.2	2	24727.59	FALSE
39988	0.3315	134254.5	2	44505.367	TRUE

The IMP_<SUBWATERSHED>_DEPTABLE_B, IMP_<SUBWATERSHED>_DEPTABLE_-C, IMP_<SUBWATERSHED>_DEPTABLE_D, and IMP_<SUBWATERSHED>-_DEPTABLE_E TABLES are used by PRINET to develop the MODEL tables for the corresponding restoration scenarios. For example, for the EDMORE subwatershed, the tables MODEL_EDMORE_B, MODEL_EDMORE_C, MODEL_EDMORE_D and MODEL_-EDMORE_E would be created. These B, C, D, E and tables are identical to the original MODEL_EDMORE_A table except in the fields that pertain to areas and volumes.

These additional B through E models have the same lakes, inflow and outflow ID's and initial water volumes as the A model. In other words, all the user input parameters that are in the A model WSHED table (MODEL_<SUBWATERSHED>_A_WSHEDS) are also copied to the B, C, D and E versions of the MODEL table.

A.2.4. Other inputs used by the subwatershed models

There were numerous additional data sources used by PRINET to run the subwatershed model calibrations / simulations:

A.2.4.1. Future Simulation Sequences

For the purposes of future simulations (water year 2001 and beyond), 11 simulation climate sequences were created. In each climate sequence, data from actual climatic years from 1980-1999 was substituted into the years 2001-2020 for 10 of the climate sequences (these sequences are labeled 001 through 010) and data from waters years 1993-1999 was substituted into the years 2001-2035 for one climate sequence (the "WET" climate sequence).

The following table shows which actual climatic water years were used in each simulated water year for each climatic sequence simulation:

Table A-1. Climatic Water Years Used in Each Simulated Water Year

Simulated Water Year	CLIMATE SEQUENCE									
	001	002	003	004	005	006	007	008	009	010
2001	1981	1994	1998	1992	1980	1999	1986	1989	1982	1990
2002	1983	1999	1997	1995	1986	1989	1999	1989	1999	1988
2003	1991	1995	1982	1985	1983	1985	1994	1996	1996	1997
2004	1995	1986	1998	1995	1982	1991	1996	1982	1988	1997
2005	1985	1981	1984	1996	1997	1980	1999	1982	1989	1996
2006	1986	1981	1990	1986	1991	1992	1995	1991	1991	1994
2007	1997	1987	1999	1988	1994	1989	1983	1997	1995	1988
2008	1994	1992	1983	1995	1988	1995	1989	1982	1986	1984
2009	1999	1982	1986	1985	1987	1997	1986	1994	1992	1986
2010	1993	1985	1995	1998	1997	1980	1990	1991	1982	1988
2011	1998	1986	1989	1994	1998	1986	1987	1989	1997	1981
2012	1983	1986	1990	1982	1998	1991	1999	1987	1984	1987
2013	1980	1997	1994	1991	1986	1991	1983	1985	1994	1986
2014	1989	1989	1990	1991	1997	1984	1982	1995	1992	1985
2015	1983	1987	1989	1994	1993	1991	1987	1983	1997	1994
2016	1996	1994	1996	1994	1994	1989	1985	1982	1986	1987
2017	1994	1993	1987	1980	1992	1987	1996	1987	1990	1985
2018	1997	1995	1999	1986	1994	1999	1995	1988	1992	1999
2019	1998	1998	1986	1984	1982	1989	1986	1984	1981	1996
2020	1986	1992	1991	1998	1998	1984	1989	1985	1987	1985

For the "WET" simulation, the climatic water years from 1993-1999 are repeated. The sequence is as follows:

Table A-2 Climatic Water Years Used For WET Simulation

WET Climate Sequence	
Simulated Water Year	Climatic Water Year
2001	1993
2002	1994
2003	1995
2004	1996
2005	1997
2006	1998
2007	1999
2008	1993
2009	1994
2010	1995
2011	1996
2012	1997
2013	1998
2014	1999
2015	1993
2016	1994
2017	1995
2018	1996
2019	1997
2020	1998
2021	1999
2022	1993
2023	1994
2024	1995
2025	1996
2026	1997
2027	1998
2028	1999
2029	1993
2030	1994
2031	1995
2032	1996
2033	1997
2034	1998
2035	1999

The actual substitution list used to relate the simulated water years to the climatic water years is contained in the table SimulSeqs, in a different format. Here are a few sample records from that table:

Sequence	SimWatYear	ClimaticWatYear
001	2001	1981
001	2002	1983
001	2003	1991
001	2004	1995
001	2005	1985
001	2006	1986
001	2007	1997
001	2008	1994
001	2009	1999

Examples will illustrate how the substitution works. For sequence 001, simulated water year 2001 uses data from climatic water year 1981. This means that from 10/1/2000 through 9/30/2001 (water year 2001), simulation 001 will use climatic data from 10/1/1980 to 9/30/981 (water year 1981). The simulation date of 10/15/2000, for example, would use climatic data from 10/15/1980. The simulation date of 4/25/2001 would use climatic data from 4/25/1981.

For this same 001 sequence, simulated water year 2002 uses climatic data from water year 1983. So 10/1/2001 (the first day of water year 2002) would use climatic data from 10/1/1982 (the first day of water year 1983).

All of this date substitution information is also contained in the precipitation tables, described in the following section.

A.2.4.2. Precipitation and Snowmelt

Each subwatershed model has at least one precipitation gage table. Each subbasin in MODEL_<SUBWATERSHED>_<RESTSCENARIO>_WSHEDS table is assigned to a certain precipitation gage. For example, here are the first few fields from a couple of records (each record corresponds to one subbasin) from the MODEL_MAUVAIS_A_WSHEDS table:

SubBSeq	Wshid	DSSubBSeq	PrecipGage	Region	InitWaterM3	InputStreamTable
675	359	690	ROLLA1	3	0	
676	1296	697	CANDO1	4	0	

For this example, SubBSeq = 675 has a precipitation gage of ROLLA1, while SubBSeq = 676 has the precipitation gage CANDO1 assigned to it. These identifiers, "ROLLA1" and "CANDO1" refer to values found in the PrecipGage field in the PrecipGages table. The PrecipGages table provides a relation between the identifier and the Access table where the precipitation, temperature, and Snowmelt data for the gage are located.

Here is the "PrecipGages" table:

PrecipGage	TableName
BELCOURT	PrecipAndTempBelcourtSim
CANDO1	PrecipAndTempCando1Sim
CHURCHF	PrecipAndTempChurchfSim
EDMORE	PrecipAndTempEdmoreSim
LEEDS	PrecipAndTempLeedsSim
ROLLA1	PrecipAndTempRolla1Sim

Note that records can be added or deleted to this table if additional precipitation and temperature information is obtained and converted for use by PRINET. For the CANDO1 precipitation gage, for example, the Access table where the information is stored is called PrecipAndTempCando1Sim. The tables listed in the "TableName" field of the above tables are all the tables in Access where the precipitation, temperature, and Snowmelt data is contained.

The precipitation gage tables contain the information for measured precipitation and temperature, as well as all the information for the future simulations. All measured values are listed under Sequence = "COMP." Here are some sample records from PrecipAndTempCando1Sim:

Sequence	SimulDate	ClimDate	TAvg	Precip	SnoPlusRain
COMP	7/25/86	7/25/86	63.39	0	0
COMP	7/26/86	7/26/86	68.695	0.18	0.18
COMP	7/27/86	7/27/86	68.51	0.38	0.38
COMP	7/28/86	7/28/86	66.29	0.16	0.16
COMP	7/29/86	7/29/86	70.385	0.46	0.46
COMP	7/30/86	7/30/86	69.83	0	0
COMP	7/31/86	7/31/86	62.775	0	0

These records show the Simulation Date (which is the same as the Climatic Date for all records labeled "COMP"), the average temperature (in the TAvg field, in degrees Fahrenheit), the Precipitation (in the Precip field, which is in inches as recorded by the gage) and a column called SnoPlusRain. The SnoPlusRain column is the calculated Rain + Snowmelt runoff, in inches. This is calculated by PRINET; the methodology to calculate it is explained in a subsequent section.

For future simulations, the Sequence field of each precipitation gage table contains the simulation climate sequence name. For example, for climate sequence 007, from the PrecipAndTempCando1Sim table, here are some sample records:

Sequence	SimulDate	ClimDate	TAvg	Precip	SnoPlusRain
007	3/27/16	3/27/85	30.695	0.01	0
007	3/28/16	3/28/85	32.345	0.07	0
007	3/29/16	3/29/85	20.16	0.41	0
007	3/30/16	3/30/85	18.46	0	0
007	3/31/16	3/31/85	21.705	0	0
007	4/1/16	4/1/85	28.25	0	0
007	4/2/16	4/2/85	40.39	0	0.4473
007	4/3/16	4/3/85	40.23	0	0.0427

For example, the simulation date 3/27/16 (3/27/2016) has a climatic date of 3/27/85, that is it uses the climatic data from 3/27/85. This climatic data source date was determined by the future simulation climate sequence for sequence 007, which dictates that simulation water year 2016 will use climatic data from water year 1985. For this climatic date, for the CANDO1 gage record, the average temperature was 30.695 and the precipitation was 0.01.

A.2.4.2.1. Generation of the SnoPlusRain column in the Precipitation and Snowmelt Data Tables

As explained previously, the average temperature and precipitation records for the Precipitation and Snowmelt Data tables are obtained by the use of the original gage records and by the use of the simulation substitution dates to generate records for the future simulations. However, the snowmelt plus rain is calculated by PRINET, and not obtained directly from the original gage records. It is this snowmelt plus rain, contained in the SnoPlusRain column, that is used by PRINET as the daily "precipitation."

The snowmelt plus rain column (SnoPlusRain, shown in inches), is calculated by PRINET using the SnoPlusRain form in the database. This was done once for each gage, before the calibrations or future simulations were run.

The algorithm to generate the snowmelt plus rain is identical to the degree-day method used in HEC-1, the U.S. Army Corps of Engineers Hydrograph Package (HEC-1 User's Manual, U.S. Army Corps of Engineers, 1990, p. 14 and p. A-43).

To calculate the snowmelt, the Snowmelt routine cycles through each day, from the first day of each simulated run to the last. For each day the routine does the following:

(1) Rainfall is calculated: If the average temperature is less than or equal to the Freezing Temperature plus 2 degrees Fahrenheit (Freezing Temperature was set at 34 degrees for all gages), then any precipitation falls as snow and goes into the snow pack as a snow-water equivalent. If the average temperature is above Freezing Temperature + 2 degrees Fahrenheit, then any precipitation falls as rain and is added to the "SnoPlusRain" column.

(2) Snowmelt is calculated: If the average temperature is above the Freezing Temperature, then snowmelt equals [(the day's average temperature) – (Freezing Temperature)] * Factor.

For all the gages used in six subbasins modeled, the freezing temperature was 34 and the factor was 0.07. This freezing temperature as well as the factor of 0.07 (called "coefficient" in the HEC-1 documentation) are within the recommend ranges detailed in the HEC-1 documentation.

Here is an example from one of the simulated gage records:

Sequence	SimulDate	ClimDate	TAvg	Precip	SnoPlusRain
008	10/27/2006	10/27/1996	35.5	0.85	0.105
008	10/28/2006	10/28/1996	34.5	0	0.035
008	10/29/2006	10/29/1996	42	0	0.56
008	10/30/2006	10/30/1996	27	0	0
008	10/31/2006	10/31/1996	14.5	0	0
008	11/1/2006	11/1/1996	16.5	0	0
008	11/2/2006	11/2/1996	17	0	0
008	11/3/2006	11/3/1996	36.5	0	0.15

On 10/25/2006, 0.85 inches of precipitation falls. Since TAvg <= 36, it falls as snow. The snow-water equivalent is now 0.85 inches (there is no snow-water equivalent from previous days since these days were warm enough to melt any accumulated snow). Since TAvg > 34, there is snowmelt. The snowmelt is (35.5 – 34) * 0.07 = 0.105 inches. This is entered into the SnoPlusRain column for that day, and the snow-water equivalent is reduced by this amount, it is now 0.85 inches – 0.105 inches = 0.745 inches. The following day, on 10/28/2006, TAvg = 34.5 > 34, so the snowmelt is (34.5 – 34) * .07 inches = 0.035 inches, and the snow-water equivalent is reduced to 0.745 inches – 0.035 inches = 0.71 inches. On 10/26/2006, TAvg = 42 > 34, so the snowmelt is (42 – 34) * 0.07 inches = 0.56, and the snow-water equivalent is reduced to 0.71 inches – 0.56 inches = 0.15 inches. All subsequent days until 11/3/2006 are sub-freezing and have no precipitation. On 11/3/2006, the temperature is 36.5 degrees, which permits a snowmelt of (36.5 – 34) * 0.07 = 0.175 inches. The snow-water equivalent is only 0.15 inches, however, so all the snow melts and 0.15 is recorded in the SnoPlusRain column, and the snow-water equivalent to be carried over to the next day is zero.

A.2.4.3. Evaporation

Evaporation was calculated based primarily on the monthly evaporation data for Devils Lake provided for 1980 through 1999 by Aldo V. Vecchia of the USGS. This source data is shown in the following table, where the values in the table are evaporation in inches:

Table A-3. Monthly Evaporation Values For Devils Lake Provided by USGS (in inches)

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1980	0	0	0	0.45	4.18	6.1	7.05	5.34	3.7	2.18	1.58	0
1981	0	0	0	2.1	3.5	4.55	5.66	6.03	4.86	1.43	1.04	0
1982	0	0	0	0.45	2.86	4.64	5.23	5.93	4.79	1.41	1.02	0
1983	0	0	0	0.45	3.59	5.09	5.5	6.41	5.7	1.68	1.21	0
1984	0	0	0	1.37	3.87	4.45	6.06	6.67	5.34	1.58	1.14	0
1985	0	0	0	1.42	4.03	5.92	6.21	5.61	4.48	1.32	0.95	0
1986	0	0	0	2.68	3.51	7.19	4.72	6.21	3.76	1.11	1.74	0
1987	0	0	0	1.31	3.63	7.28	4.92	6.09	4.32	3.82	1.84	0
1988	0	0	0	1.44	4.12	8.81	6.78	6.63	4.61	4.08	1.97	0
1989	0	0	0	1.33	3.72	5.08	6.9	6.94	5.24	4.64	2.24	0
1990	0	0	0	1.36	3.81	5.44	5.82	6.76	5.89	5.21	2.51	0
1991	0	0	0	1.51	4.39	4.97	5.87	6.97	4.95	4.38	1.06	0
1992	0	0	0	1.37	3.86	5.66	3.98	5.26	4.41	3.9	0.94	0
1993	0	0	0	1.26	3.44	4.31	3.43	4.49	3.92	3.47	0.84	0
1994	0	0	0	1.34	3.75	4.55	4.88	5.64	4.73	2.79	0.67	0
1995	0	0	0	2.11	4.58	5.16	6.25	6.21	5.08	2.99	0.72	0
1996	0	0	0	0.45	2.7	5.08	5.02	6.12	4.55	2.69	0.65	0
1997	0	0	0	0.45	3.52	7.02	4.75	5.65	5.02	2.96	0.71	0
1998	0	0	0	2.08	3.46	4	5.6	6.05	5.56	3.28	0.79	0
1999	0	0	0	0.45	3.14	4.42	4.91	5.56	5.59	3.3	0.79	0

This data appears in the database in a different format, in the table DevilsLakeUSGS. Here is a sample of a few records of that table:

Year	Month	USGSEvapIn
1980	1	0
1980	2	0
1980	3	0
1980	4	0.45
1980	5	4.18
1980	6	6.1
1980	7	7.05
1980	8	5.34
1980	9	3.7
1980	10	2.18
1980	11	1.58
1980	12	0
1981	1	0
1981	2	0

Because PRINET requires daily evaporation values, these monthly evaporation values provided by the USGS had to be converted into daily values (they were also converted from inches to

meters). The daily evaporation values for use by PRINET are contained in the EvapSim table. Following are a few sample records from that table:

Sequence	SimulDate	ClimDate	EvapPropM
COMP	8/1/1982	8/1/1982	0.005416202
COMP	8/2/1982	8/2/1982	0.00644786
COMP	8/3/1982	8/3/1982	0.003868716
COMP	8/4/1982	8/4/1982	0.005932031
COMP	8/5/1982	8/5/1982	0.005416202
COMP	8/6/1982	8/6/1982	0.007737432
COMP	8/7/1982	8/7/1982	0.004900374
COMP	8/8/1982	8/8/1982	0.006963688
COMP	8/9/1982	8/9/1982	0.00644786
COMP	8/10/1982	8/10/1982	0.004900374
COMP	8/11/1982	8/11/1982	0.004384545
COMP	8/12/1982	8/12/1982	0.006963688

The Sequence column is the simulation climate sequence number. When Sequence = COMP, the evaporation values (in the EvapPropM field) are the historical values, in meters, for evaporation. For other sequences, they are the simulated daily evaporations. For example, here is another section of the EvapSim table:

Sequence	SimulDate	ClimDate	EvapPropM
002	8/1/2009	8/1/1982	0.005416202
002	8/2/2009	8/2/1982	0.00644786
002	8/3/2009	8/3/1982	0.003868716
002	8/4/2009	8/4/1982	0.005932031
002	8/5/2009	8/5/1982	0.005416202
002	8/6/2009	8/6/1982	0.007737432
002	8/7/2009	8/7/1982	0.004900374
002	8/8/2009	8/8/1982	0.006963688
002	8/9/2009	8/9/1982	0.00644786
002	8/10/2009	8/10/1982	0.004900374
002	8/11/2009	8/11/1982	0.004384545
002	8/12/2009	8/12/1982	0.006963688

These show the evaporation records for climate sequence 002, for simulated dates 8/1/2009 through 8/12/2009. Because the climatic dates used for these dates are 8/1/1982 through 8/12/1982 (based on climate sequence 002 which has simulated water year 2009 = climatic water year 1982), the evaporation numbers are the same as in the previous table.

The daily values were developed using the proportional weights from a nearby evaporation gage, the Langdon gage. These proportional weights were applied to the USGS evaporation values. For example, the month of August 1982 had a total evaporation of 5.93 inches according the USGS evaporation values. The Langdon evaporation provided a daily distribution of

evaporation values for the month of August 1982. Therefore each day in August 1982 is assigned a weighting value equal to (Langdon Evaporation for that Day) / (Total Langdon Evaporation for August 1982). These daily weights were multiplied by the USGS evaporation values to provide daily values for use by PRINET.

The evaporation values for the Langdon gage are in the table LangdonDailyEvap. Following are a few sample records from that table, with the evaporation shown in inches:

Year	Month	Day	Evaporation
1982	8	1	0.21
1982	8	2	0.25
1982	8	3	0.15
1982	8	4	0.23
1982	8	5	0.21
1982	8	6	0.3
1982	8	7	0.19
1982	8	8	0.27
1982	8	9	0.25
1982	8	10	0.19
1982	8	11	0.17
1982	8	12	0.27

For example, the total Langdon evaporation for August 1982 turns out to be 5.84 inches. For 8/6/1982, then, the weighting factor would be 0.3 / 5.84 = 0.005137. This is multiplied times the USGS evaporation value for August 1982 of 5.93 inches: 0.005137 * 5.93 = 0.304623 inches. Converted to meters, 0.304623 inches * 0.0254 meters / inch = 0.00077374 meters. This is the evaporation value used for climatic day 8/6/1982 by PRINET.

Evaporation values for water years 1978 and 1979 (used in the warm-up period before the 1985-1999 calibration period), as well as for water year 2000 (used to warm up for the simulation runs) were not provided by the USGS. For water years 1978 and 1979, the same evaporation was used as was generated for water year 1980. For water year 2000, the evaporation was estimated based on the Langdon evaporation records and on the evaporation values provided by the USGS for 1980-1999. The evaporation at the Langdon gage was substantially higher in water year 2000 than in 1999. Based on comparisons between the behavior of the Langdon gage values versus the USGS evaporation values, a value of 35.97 inches of evaporation was estimated for water year 2000. This compares to a range of about 26-38 inches per water year from the USGS evaporation values for 1980-1999.

A.2.4.4. Lake data

Lakes are a feature of the program which allows the use of storage-outflow curves for specified water bodies. The normal behavior of on-river depressions in PRINET is that water flows into a subbasin's on-river depressions until their depressional capacity is exceeded. At that point any excess immediately flows to the downstream subbasin. Lakes override this behavior by using a

storage-outflow table. The storage-outflow relationships for all the lakes used in the different subbasins are shown in the following table (the field names are rotated vertically to fit the table on the page):

LakeID	LAKE	LakeElevationFT	VolumeM3	VolumeAcreFeet	FlowM3PD	FlowCFS	AreaM2	AreaAcres
1	SWMORR	1452	0	0	0	0	0	0
1	SWMORR	1453	1480188	1200	0	0	5058575	1250
1	SWMORR	1454	3947168	3200	0	0	9307778	2300
1	SWMORR	1455	7647638	6200	0	0	12949950	3200
1	SWMORR	1456	13321690	10800	0	0	16592130	4100
1	SWMORR	1457	19489140	15800	0	0	19829610	4900
1	SWMORR	1458	25903290	20999	0	0	22662420	5600
1	SWMORR	1459	33304230	26999	0	0	25495220	6300
1	SWMORR	1460	37577230	30463	168812	69	27012790	6675
1	SWMORR	1460	41938660	33999	479524	196	28530360	7050
1	SWMORR	1461	46833220	37967	880759	360	30047940	7425
1	SWMORR	1461	51806580	41999	1355390	554	31565510	7800
1	SWMORR	1462	57073420	46268	1896078	775	33184250	8200
1	SWMORR	1462	62907990	50998	2490590	1018	34803000	8600
1	SWMORR	1463	68297820	55368	3138927	1283	36320570	8975
1	SWMORR	1463	74379450	60298	3836194	1568	37838140	9350
2	DRYLAKE		0	0	0	0	0	0
2	DRYLAKE	1442	616745	500	0	0	7689034	1900
2	DRYLAKE	1443	3453772	2800	0	0	12342920	3050
2	DRYLAKE	1444	7647638	6200	0	0	15378070	3800
2	DRYLAKE	1445	12951650	10500	0	0	18210870	4500
2	DRYLAKE	1446	18502350	15000	73397	30	20638990	5100
2	DRYLAKE	1447	25163200	20399	342517	140	22257730	5500
2	DRYLAKE	1448	28616970	23199	562707	230	22864760	5650
2	DRYLAKE	1448	32070740	25999	831828	340	23471790	5800
2	DRYLAKE	1449	39471680	31999	1492397	610	24685850	6100
2	DRYLAKE	1450	47119320	38199	2348690	960	25899900	6400
2	DRYLAKE	1451	54766960	44399	3400708	1390	27113960	6700
2	DRYLAKE	1452	62907990	50998	3938950	1610	28328020	7000
3	MIKESLAKE		0	0	0	0	0	0
3	MIKESLAKE	1442	863443	700	0	0	1687541	417
3	MIKESLAKE	1442	1110141	900	293586	120	2448350	605
3	MIKESLAKE	1443	1480188	1200	513776	210	4358468	1077
3	MIKESLAKE	1444	2960376	2400	1027552	420	6114805	1511
3	MIKESLAKE	1445	4440564	3600	1345604	550	7873166	1945
3	MIKESLAKE	1446	7154242	5800	1663656	680	9210653	2276
3	MIKESLAKE	1447	11101410	9000	1932776	790	10550160	2607

LakeID	LAKE	LakeElevationFT	VolumeM3	VolumeAcreFeet	FlowM3PD	FlowCFS	AreaM2	AreaAcres
4	CHAINLAKE		0	0	0	0	0	0
4	CHAINLAKE	1440	666085	540	70950	29	1078488	266
4	CHAINLAKE	1442	1726886	1400	269121	110	2804474	693
4	CHAINLAKE	1442	2096933	1700	623871	255	3237488	800
4	CHAINLAKE	1443	3237911	2625	1228169	502	4313953	1066
4	CHAINLAKE	1444	5057309	4100	2162752	884	6814912	1684
4	CHAINLAKE	1445	10176290	8250	3395815	1388	9313848	2301
4	CHAINLAKE	1446	16035370	13000	4932250	2016	10327590	2552
5	LAKEALICE	1438	0	0	0	0	0	0
5	LAKEALICE	1439	481061	390	755985	309	3156551	780
5	LAKEALICE	1440	1887240	1530	1370069	560	6070290	1500
5	LAKEALICE	1441	4169196	3380	2260615	924	8903092	2200
5	LAKEALICE	1442	7326931	5940	3288167	1344	11816830	2920
5	LAKEALICE	1443	11372780	9220	4521229	1848	14730570	3640
5	LAKEALICE	1444	16652120	13500	5754291	2352	25733980	6359
5	LAKEALICE	1445	24053060	19499	6713340	2744	36733350	9077
5	LAKEALICE	1446	34537720	27999	7809396	3192	41415560	10234
6	LAKEIRVINE	1438	0	0	0	0	0	0
6	LAKEIRVINE	1439	1541863	1250	0	0	10117150	2500
6	LAKEIRVINE	1440	5118984	4150	0	0	13354640	3300
6	LAKEIRVINE	1441	9275845	7520	0	0	14406820	3560
6	LAKEIRVINE	1442	11980890	9713	0	0	14576790	3602
6	LAKEIRVINE	1442	13784250	11175	247102	101	14690100	3630
6	LAKEIRVINE	1443	18304990	14840	587173	240	14973380	3700
6	LAKEIRVINE	1444	22899740	18564	902778	369	15175730	3750
6	LAKEIRVINE	1445	27556170	22339	1365176	558	15378070	3800
6	LAKEIRVINE	1446	45639130	36999	1849594	756	17996390	4447
6	LAKEIRVINE	1447	56740540	45999	2360923	965	20614700	5094

The preceding table is based on the tables shown in the January 1995 report “Devils Lake Basin Water Management Plan” (North Dakota State Water Commission, 1995), which provides flows, volumes and surface areas for the six lakes listed.

For each lake, there are a series of records in the Lake table. Each record corresponds to a lake elevation. For each lake elevation, there is a corresponding volume (VolumeM3, the volume in cubic meters) and Flow (FlowM3PD, the flow in cubic meters per day). The actual Lake table used by PRINET has many additional records not shown here; these are records interpolated between various elevations so that the flows will not shift abruptly. Furthermore, the Lake table has additional records to allow for higher flows when the lake volumes exceed the range given in the 1995 report. The maximum flow permitted for any lake was 50% greater than the highest

flows given in the 1995 North Dakota State Water Commission Report. These higher flows were added to the lake table because the observed flows in some years at Big Coulee were considerably higher than the maximum outflow given for Lake Irvine (the final lake before the Big Coulee outlet) in the NDSWC report.

For each lake, for each simulation day, PRINET looks up the nearest volume which is less than or equal to current water volume of the lake in the simulation, and uses the flow that is contained in that record of the Lakes table.

Note that Mike's Lake, Chain Lake, and Lake Alice have downstream dependencies; that is, their flow depends on the elevation of the immediate downstream lake (the flow sequence is Mike -> Chain -> Alice -> Irvine). These dependencies are not taken into account in PRINET, which could affect the timing of the flows from these lakes.

Lakes were assigned as to the following subwatersheds:

Lake	LakeID	Subwatershed where located
Sweetwater-Morrison Lakes	1	EDMORE
Dry Lake	2	STARKWEATHER
Mike's Lake	3	MAUVAIS
Chain Lake	4	MAUVAIS
Lake Alice	5	MAUVAIS
Lake Irvine	6	MAUVAIS

A.2.4.5. Input Streams

The Devil's Lake watershed was separated in 6 subwatersheds: Comstock, Hurricane Lake, Mauvais, Mauvais 6100, Edmore, and Starkweather. Some of these subwatersheds flow into other subwatersheds. Mauvais 6100 and Hurricane Lake flow into Mauvais. Edmore flows into Starkweather. PRINET is able to accommodate inflows by the use of input streams. Input streams are specified in the InputStreamTable field of subwatershed MODEL table. For example, Hurricane Lake is input into the Mauvais model as shown in the first few fields of a record from the MODEL_MAUVAIS_A_WSHEDS:

SubBSeq	WShid	DSSubBSeq	PrecipGage	Region	InitWaterM3	InputStreamTable
2993	3001	3006	CHURCHF	4	0	HURRLAKE

PRINET inputs the daily outflows from Hurricane Lake, using the Hurricane Lake's model output tables, into the subbasin where HURRLAKE is specified as an input stream table. The climate sequence and restoration scenario are matched, for example if restoration scenario "B" of sequence 008 is running, then OUTPUT_HURRLAKE_008_B will be the table used to retrieve the inflows into the MAUVAIS model.

A.2.4.6. Additional Tables and Queries Used by PRINET

In addition to model and input tables, there are other tables and queries that are used in the running of PRINET. These are not data but are either templates (in the case of tables) which are used to create new tables during the running of PRINET, or queries which are can be considered part of PRINET program.

Furthermore, the program itself generates various temporary tables and queries while it is running. All of these tables and queries are prefixed with the letters TEMP or Temp. Some of these tables and queries may remain in the database after the simulation is completed, but once PRINET has stopped running, these temporary tables and queries have no function and can be deleted by the user.

A.2.5. Model Outputs

PRINET generates output tables of four types: output, outflow, inflow, and season tables.

A.2.5.1. Output Tables

Output tables contain the daily flows at the outlet the subwatershed being modeled, as well as various other calculated values for each day. The following shows a sample of a few records from an output table:

Date	CMPD	StorageVolUtil	SoilCapacityFraction	PrecipM3	LostToETM3	LostToEvapM3	LostToPercM3
4/1/2005	0	0.5717619	0.601377	0	145770.3	80426.88	0
4/2/2005	0	0.5713992	0.6017377	0	145638.3	80382.82	0
4/3/2005	0	0.5710367	0.602098	0	145506.6	80338.77	0
4/4/2005	0	0.5706744	0.6024581	0	145374.9	80294.71	0
4/5/2005	0	0.5703123	0.6028178	0	145243.4	80250.66	0
4/6/2005	0	0.5699504	0.6031772	0	145111.9	80206.6	0
4/7/2005	0	0.5695887	0.6035363	0	144980.6	80162.55	0
4/8/2005	0	0.5692272	0.6038951	0	144849.4	80118.49	0
4/9/2005	0	0.5688659	0.6042535	0	144718.4	80074.44	0
4/10/2005	26828.6	0.5877758	0.5862642	11855380	151296.8	83072.66	0
4/11/2005	96627.75	0.6392586	0.5606636	23224360	160658.5	90088.36	0

4/12/2005	27265.65	0.6405806	0.5610612	0	160513.1	90169.45	0	

The fields have the following definitions:

- Date – The simulation day.
- CMPD – The outflow in cubic meters (the field name CMPD stands for Cubic Meters Per Day).
- StorageVolUtil – The fraction of the total intact depressional volume that is filled with water. Note that in the case of the restoration simulations (restoration scenarios B, C, D and E), there is a one-time abrupt change in the total intact depressional volume. This occurs on the restoration transition date, when the intact depressional volume is suddenly increased to a new, higher level.
- SoilCapFraction – The weighted average over the subwatershed of the soil capacity (i.e. dryness) as a percentage of the total potential soil capacity.
- PrecipM3 – Cubic meters of precipitation over the subwatershed.
- LostToETM3 – Cubic meters of water lost to evapotranspiration.
- LostToEvapM3 – Cubic meters of water lost to evaporation from standing water in depressions or lakes.
- LostToPercM3 - Cubic meters of water lost to percolation.

The naming convention of the output tables is:

OUTPUT_<SUBWATERSHED>_<SEQ>_<RESTSCENARIO>

OUTPUT tables are included for every combination of:

- Climate sequence 001 through 010 and the WET climate sequence.
- Restoration scenarios A through E.
- The 6 subwatersheds: COMSTOCK, MAUV6100 (everything upstream of the 05056100 stream gage), MAUVAIS (the Mauvais Coulee subwatershed, excluding the area covered by MAUV6100, but including St. Joe and Calio), EDMORE, STARKWEATHER, and HURRLAKE (Hurricane Lake subwatershed).

For example for EDMORE, climate sequence 008, restoration scenario C, the table name is OUTPUT_EDMORE_008_C.

A.2.5.2. *Outflow Tables*

Outflow tables are those generated by placing text in one or more of the ExtraOutflowID fields in the MODEL_<SUBWATERSHED>_<RESTSCENARIO>_WSHEDS table. They give daily water volume flowing into a particular subbasin. The naming convention is:

OUTFLOW_<SUBWATERSHED>_<ExtraOutflowID>_<SEQ>_<RESTSCENARIO>

For example, OUTFLOW_MAUVAIS_MIKE_003_B is the table with the outflows at Mike's Lake for the MAUVAIS subwatershed model, climate sequence 003, restoration scenario B.

The data structure of an outflow table consists only of the date and flow in cubic meters per day. Following is a sample of a few records from an outflow table.

Date	CMPD
4/9/2017	553165.9
4/10/2017	519158.8
4/11/2017	485151.7
4/12/2017	451144.6
4/13/2017	0
4/14/2017	0

An explanation of how to enter an outflow file for a model's subbasin appears in the section describing the MODEL_ tables. The following outflow tables are included for all climate sequence restoration scenarios:

OUTFLOW_MAUVAIS_05056270

OUTFLOW_MAUVAIS_05056400

OUTFLOW_MAUVAIS_MIKE

OUTFLOW_MAUVAIS_CHAIN

OUTFLOW_MAUVAIS_ALICE

OUTFLOW_MAUVAIS_IRVINE

For the calibration runs (COMP_A), all the above are included plus these additional outflows:

OUTFLOW_MAUVAIS_05056060_COMP_A

OUTFLOW_HURRLAKE_05056340_COMP_A

A.2.5.3. Inflow Tables

Inflow tables are those generated by placing text in one or more of the ExtraInflowID fields in the MODEL_<SUBWATERSHED>_<RESTSCENARIO>_WSHEDS table. They give daily water volume flowing into a particular subbasin. The naming convention is:

INFLOW_<SUBWATERSHED>_<ExtraInflowID>_<SEQ>_<RESTSCENARIO>

For example, INFLOW_MAUVAIS_MIKE_003_B is the table with the inflows at Mike's lake for the MAUVAIS subwatershed model, climate sequence 003, restoration scenario B.

The structure of inflow tables is identical to that of outflow tables. As in the outflow tables, only the fields Date and CMPD are present.

An explanation of how to request an inflow table for a model's subbasin appears in the section describing the MODEL_ tables. The following inflow tables are included for all climate sequences and restoration scenarios, and for the COMP_A run also:

INFLOW_MAUVAIS_MIKE

INFLOW_MAUVAIS_CHAIN

INFLOW_MAUVAIS_ALICE

INFLOW_MAUVAIS_IRVINE

A.2.5.4. Seasons Tables

Each simulation run produces a corresponding SEASONS table which provides that dates of the transitions between high infiltration (H or Regime 2) and low infiltration (L or Regime 1) and back.

The naming convention for the SEASONS tables is:

OUTPUT_<SUBWATERSHED>_SEASONS_<SEQ>_<RESTSCENARIO>

For example, the SEASONS table for the EDMORE subwatershed, for climate sequence 005 and restoration scenario A is:

OUTPUT_EDMORE_SEASONS_005_A

The SEASON table shows at which date the Regime (season) began for each gage used by each simulation. Here are a few sample records from a SEASONS table.

Gage	StartDate	Season	MovingAverage
CHURCHF	5/16/2018	H	45.7673263549805
CHURCHF	11/10/2018	L	34.1911659240723
CHURCHF	4/29/2019	H	45.7068328857422
CHURCHF	11/17/2019	L	34.6288375854492
CHURCHF	5/15/2020	H	45.7041702270508
EDMORE	10/1/1978	H	0
EDMORE	11/18/1978	L	34.6833343505859
EDMORE	5/28/1979	H	45.3499984741211
EDMORE	11/8/1979	L	34.7999992370605
EDMORE	4/27/1980	H	45.283332824707
EDMORE	11/11/1980	L	34.9166679382324

For this example, the entry threshold for Regime 1 (Low infiltration) was set at 35 degrees F, and the entry threshold for Regime 2 was set at 46 (in the MODEL_<SUBWATERSHED>-_<RESTSCENARIO> table). When in Regime 2 and the 30-day moving average first dipped below 35 degrees F, the Regime switched to 1, and the date and 30-day moving average temperature were recorded in the SEASON table. When in Regime 1 and the 30-day moving average first went above 45 degrees F, the Regime switched to 2, and the data and 30-day moving average temperature were recorded in the SEASON table. So for the table shown, 5/16/2018-11/09/2018 are in the High regime (Regime = 2) while 11/10/2018 to 4/18/2019 are in the Low regime (Regime = 1), for the CHURCHF gage.

Notice that for the record for the Edmore Gage at Date 10/1/1978, there is no moving average. The season for the first date of the simulation is not set by the 30-day moving average but by the start day's average temperature (which is not shown but must have been greater than 35 degrees since PRINET assigned it to the High regime [Regime = 2]).

A.3. PRINET SIMULATION FLOWCHART

A flowchart, shown in Figure A-8. Pothole-River Networked Watershed Model (PRINET) Flow Chart., illustrates the calculations performed during the simulations. Figure A-9 illustrates some of the variables used in the soil moisture accounting routines that are used by PRINET.

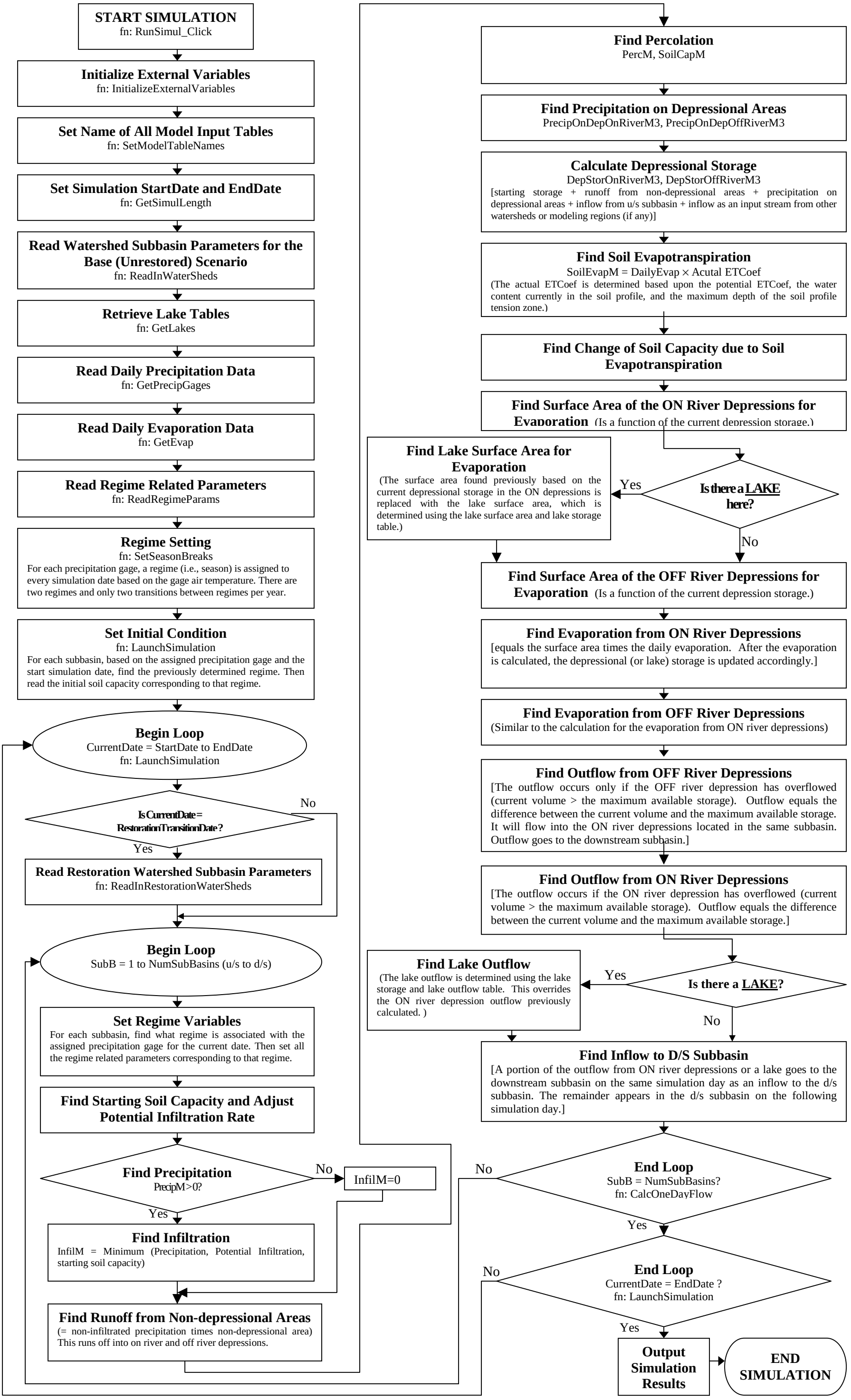

Figure A-8. Pothole-River Networked Watershed Model (PRINET) Flow Chart.

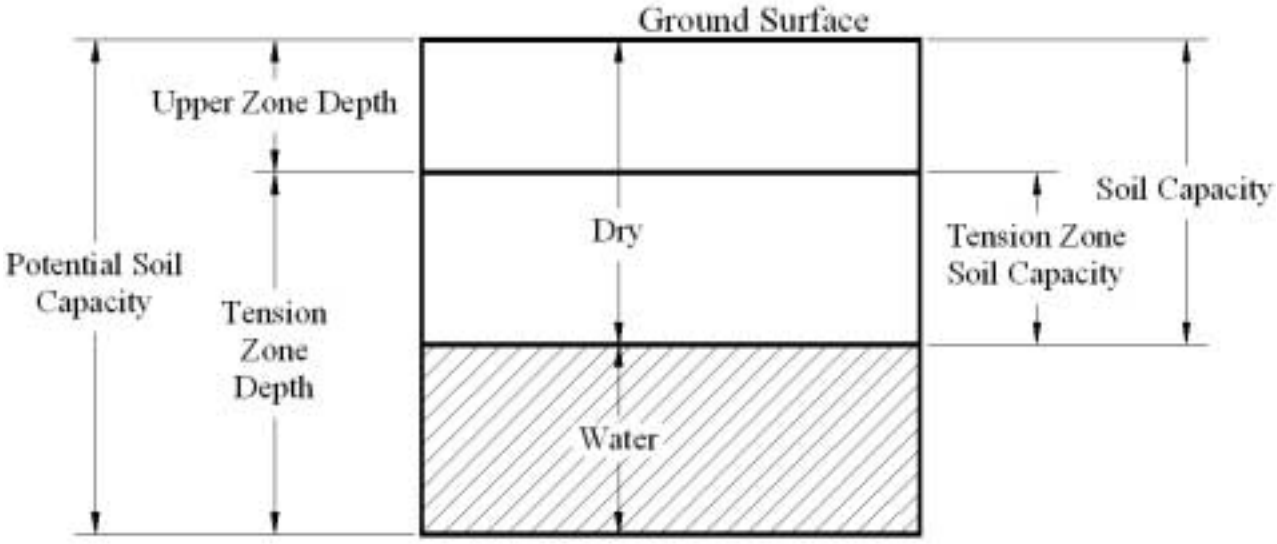

Figure A-9. A schematic representation of soil profile, with variables labeled.

All dimensions are depths of equivalent water; they do not correspond to physical depths nor do the dry / water areas correspond to physical locations. PRINET tracks soil capacity as a depth in meters for each subbasin on a daily basis. Soil capacity is decreased by infiltrated precipitation. Soil capacity is increased by evapotranspiration or percolation.

Some notes regarding the flowchart follow. The headings of the following sections correspond the titles of the boxes in the flowchart.

A.3.1. Begin Loop SubB = 1 To NumSubBasins

The subbasins are sequenced so that the lowest numbered subbasins are the most upstream. Therefore, this loop cycles from the upstream to downstream subbasins.

A.3.2. Read Restoration Watershed Subbasin Parameters

When the simulation date reaches the restoration transition date (the date at which restoration of drained depressions is assumed to be implemented) which was 10/1/2003, PRINET reads the parameters from the MODEL table corresponding to the restoration scenario being simulated. These parameters are used for all simulations from 10/1/2003 forward. For example, suppose the current simulation is for the COMSTOCK watershed, climate sequence 005, restoration scenario C. PRINET begins the simulation on 10/1/1978, but using the geometric information for COMSTOCK contained in MODEL_COMSTOCK_A. It is not until the simulation date reaches 10/1/2003 that PRINET reads in the parameters from MODEL_COMSTOCK_C and begins to use those parameters for the simulations.

A.3.3. Find Starting Soil Capacity and Adjust Potential Infiltration Rate

The actual infiltration rate (the amount of precipitation that can permeate into the soil on a given day) for a subbasin for a simulation day is based on the maximum potential infiltration rate (from the InfilMPD field in the REGIME table, this changes seasonally), adjusted according to how saturated the soil is. Potential infiltration is adjusted as follows:

Actual potential infiltration = maximum potential infiltration * (soil capacity / potential soil capacity)

Since soil capacity, which is re-computed daily, is a measure of dryness (how much more water the soil could hold), actual potential infiltration equals its maximum value when the soil is completely dry. As the soil capacity decreases (i.e. the soil becomes more saturated), the actual potential infiltration is reduced.

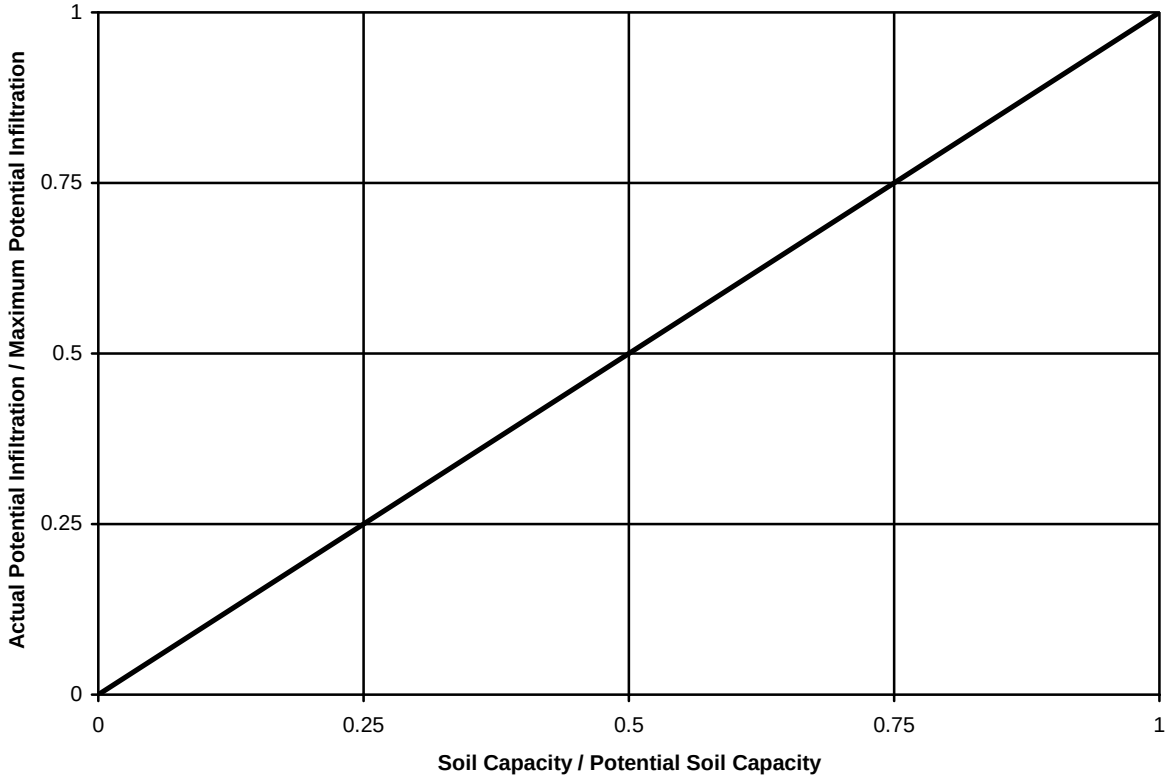

Figure A-10. Relationship between soil capacity and actual potential infiltration

A.3.4. Find Percolation

Percolation is loss of soil moisture to groundwater. Percolation, if it occurs, increases the soil capacity by the percolation amount. Percolation is only permitted by PRINET if the soil capacity on a given day is less than the upper zone depth (the upper zone depth is a parameter specified in the REGIME table). Since the upper zone is generally not very deep, this only occurs when the

soil is very saturated. If the soil capacity is less than the upper zone depth, the percolation occurs at the rate specified in the REGIME table.

A.3.5. Find Runoff from Non-depressional Areas

After the actual potential infiltration rate for a subbasin is calculated, then if the precipitation amount for the day exceeds that rate, all excess (the excess of precipitation minus infiltration) runs off into the depressions of the subbasin. Since on-river and off-river depressions are accounted for separately,

Off-river non-depressional runoff = (precipitation not infiltrated) * contributing area for off-river depressions

and

On-river non-depressional runoff = (precipitation not infiltrated) * contributing area for on-river depressions.

These runoffs are added, respectively, to the off-river and on-river depressional storage for that subbasin. Note that the contributing areas do not include the surface areas of intact depressions, precipitation on these is accounted for separately in PRINET.

A.3.6. Find Soil Evapotranspiration

Each simulation day has a pre-set evaporation amount, read from the EvapSim table. This daily value, multiplied times the evapotranspiration coefficient (ET Coefficient, from the Regime table) is the potential evapotranspiration, or potential ET.

Potential ET = Evaporation * ET Coefficient

“Evaporation” in the equation above may refer to a pan evaporation value, or an open water value. For this study, daily Devils Lake evaporation values were used. These daily values were computed by converting the monthly Devils Lake evaporation data provided by the USGS to daily values using the pattern observed at the Langdon pan evaporation gage.

Soil evapotranspiration is calculated based on the following graph, which has been adapted from Bennett (1998). It is the method used by HEC-HMS to calculate evapotranspiration:

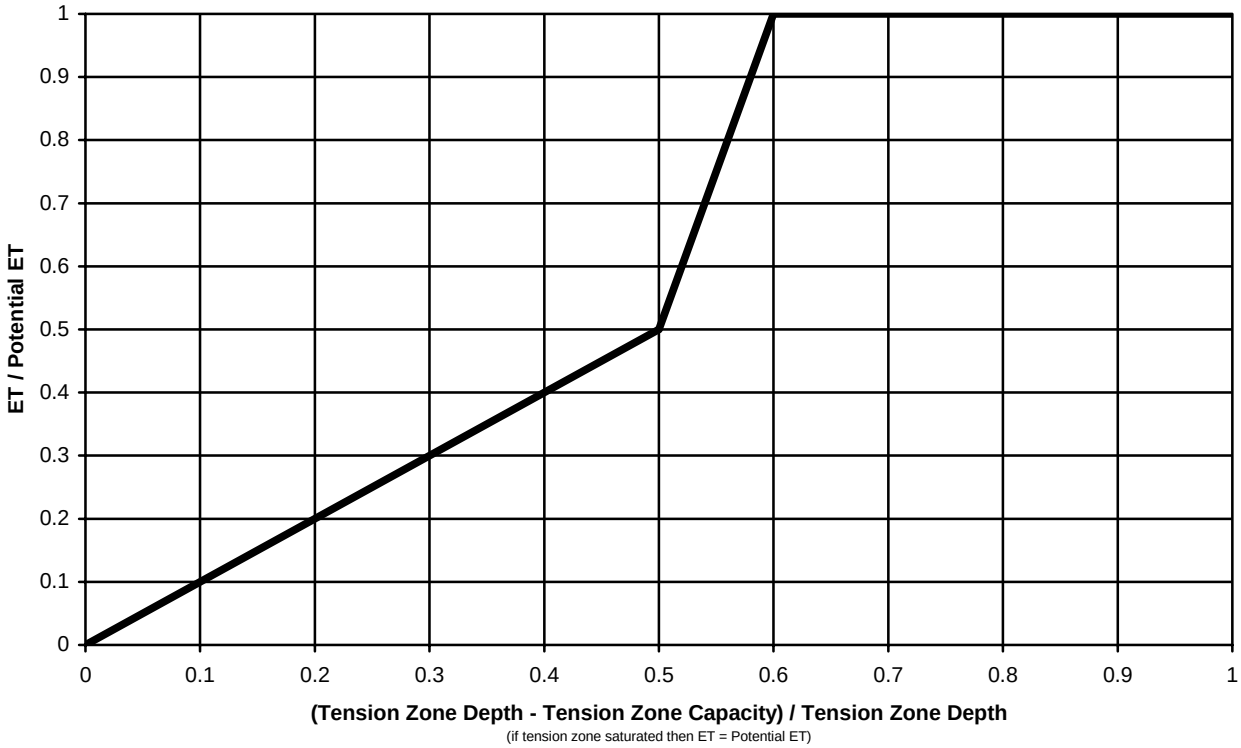

Figure A-11. Graph showing how actual ET is calculated based on Tension Zone Capacity

For example, the Tension Zone Capacity is 0.27 m, and if on a particular simulation day in a particular subbasin, Tension Zone Depth is 0.13 m, the x-axis value for the above graph is:

$(0.27 – 0.13) / 0.27 = 0.518.$

Reading from y-axis of the graph, ET/Potential ET = 0.59. So for this day and location, (actual) ET = 0.59 * Potential ET = 0.59 * ET Coefficient * Evaporation. The soil capacity (a measure of soil dryness) is increased by this amount, with the limitation that the soil capacity cannot exceed the potential soil capacity.

A.3.7. Find Precipitation on Depressional Areas

Precipitation directly over intact depressional areas is assumed to contribute directly to the water volume in the depression. Each day's precipitation volume that contributes directly to each subbasin's depressional water volume equals the depressional area times the day's precipitation. On-river and off-river depressions are accounted for separately.

This is in contrast to the non-depressional areas, for which possible infiltration and runoff is calculated daily. In essence, the intact depressions are assumed to have impermeable bottoms.

A.3.8. Find Surface Area of the On-river Depressions for Evaporation

As the volume of water in depressions decreases, so does the surface area available for evaporation. The relationship used is:

Surface Area For Evaporation = Depressional Surface Area When Full * (Current Volume of Water / Depressional Volume) ^ 0.5

For example, if the depression was 40% full, then the surface area for evaporation would be (0.4) ^0.5 = 0.632 times the depressional surface area when full. The daily evaporation data is applied to this surface area to determine the volume of evaporation. This is done daily by subbasin. Since each subbasin's depressional volume is grouped by on-river and off-river depressions, the evaporation calculation is applied separately to each.

A.3.9. Find Outflow from On-river Depressions

Outflow from on-river depressions occurs when the depressional water storage exceeds the depressional volume (or in the case of a lake, flow occurs according to the storage-outflow relationship contained in the Lakes table).

APPENDIX A-2

PRINET Source Code

```
'Pothole-River Networked Watershed Model V1.0 (PRINET)

'Variables in this program that have units contain the units in the name as
'capitial letters, usually at the end of the variable name.  The following
'units are used:
'M - meters
'M2 - square meters
'M3 - cubic meters
'MPD - meters per day
'M3PD or CMPD - cubic meters per per day
'F - degrees Farenheit
'
'Any variable with the word Date in it is a calendar date.
'Other variables without units are either identifiers or dimensionless.

'The term "program" refers to this code and to fixed tables and queries
'used in conjuction with the running of this code.
'The term "model" refers to the specific model whose data is loaded in
'and for which the simulation is performed.

Option Explicit 'forces variable declaration in this module

'WaterShedType is a type is used to contain the information for each watershed
'This information is fixed during the simulation, until the date of of restoration
'transition takes arrives.  At that some of the depressional volumes, areas, and watershed
'areas may be changed once and left at the new values until the end of the simulation
'This type will be used for the externally declared WS() array where
'each element in this array corresponds to a subbasin in the model
'Except as noted, all variables are read from the record of identical name in the
'WSHEDS table of the model.
Private Type WaterShedType
    SubBSeq As Long 'The Subbasin sequence number assigned by this program to each subbasin
    WShid As Long 'The Watershed ID number from Arcview
    DSSubBSeq As Long 'The subbasin sequence number of the subbasin downstream of this
subbasin
    WSAreaNonDepOnRiverM2 As Double 'The contributing area of
        'all intact and drained depressions that intersect the HEC-GeoHMS River network.
It does not
        'include the surface area of intact depressions.

    DepVolOnRiverM3 As Double 'The depressional volume of all depressions that intesect the
HEC-GeoHMS
        'river network

    DepAreaOnRiverM2 As Double 'The surface area (when full) of all intact depressions that
intersect
        'the HEC-GeoHMS River network

    WSAreaNonDepOffRiverM2 As Double 'The contributing area of all intact and drained
        'depressions that intersect the HEC-GeoHMS river network.

    DepVolOffRiverM3 As Double 'The depressional volume of all intact depression
        'that do not intersect the HEC-GeoHMS river network

    DepAreaOffRiverM2 As Double 'The surface area (when full) of all intact depressions
        'that do not interesect the HEC-GeoHMS river network

    WSAreaNonDepFractionOfTotal As Double 'The fraction of the subwatershed that is not
        'the surface area of intact depressions.  This variable is calculated by the
program
        'after the watershed information is read-in, and is used in CalcOneDayFlow
        'to weight the soil capacities of each subwatershed to arrive at a weighted average
        'soil capacity for the entire watershed

    InitWaterM3 As Double 'The initial volume of water in cubic meters that is held in the
        'intact depressions of the wateshed.  This is used at the start date of the
simulation
```

```
    PrecipGage As String 'The string identifier of the precipitation gage to use for this
        'subwatershed.

    Region As Long 'The region number of this subwatershed.  This number is refered to
        'in the REGIMES table for the model

    InputStreamTable As String 'The name of the table that provides an input stream into
        'this subwatershed.  Those tables's names have the format
        'OUTPUT_<InputStreamTable>_<Simulation>_<Restoration Level>
        'e.g. OUTPUT_DRYCREEK_COMP_A.
        'The fields from the input stream table that are used are [Date] and [CMPD] (cubic
meters
        'per day.  The output file of any model can function as an input stream into
another
        'model.

    LakeID As Integer 'The ID number of the lake that exists in this subwatershed.  These
        'lakes are referenced in the LAKES table

    ExtraInflowID As String 'Name for file to write the INFLOWS into this subwatershed
        'These inflows will include all upstream inflows, but will not include any input
        'stream that was routed into this subwatershed

    ExtraInflowIndex As Integer 'An index/ID number for this file.  This is generated by
the program
        'If = 0 there is no output file for this subwatershed

    ExtraOutflowID As String 'Name for file to write the OUTFLOWS of this subwatershed

    ExtraOutflowIndex As Integer 'An index/ID number for this file.  This is generated by
the program
        'If = 0 there is no Outflow output file for this subwatershed

    VBInputStreamID As Integer 'An ID number for any input stream into the subwatershed.
        'This is generated by the program.

    VBGageID As Integer  'An ID number for the gage assigned to this subwatershed.

End Type

'The following Type is used to hold the daily information of the each subbasin's
'status during the runs, and to tract outflows and inflows during the run.
'This Type will be used to declare the external ST() array.  This array functions
'by storing the the current date's information in the row.  This array only has
'three rows, numbered 0, 1 and 2.  The dates will cycle through each of these three
'rows.  For example the simulation might start at 0 (although it might start at 1
' or 2), then the next date would be 1,
'the 2nd day would be 2, and the 3rd day would be back to 0.  The formula for
'determining the row is CurrentDate - MOD STArrayDim1, where STArrayDim1
'is set to 3.  This takes the remainder of the simulation day number (starting
'at 0 for the first day) divided by 3.
'Before each day's run, the ST() row that is 2 days away is initialized.
Private Type SimulTraceType
    SimulDate As Date 'The simulation date which corresponds to this row
    StartDepStorOnRiverM3 As Double 'The water stored in intact depressions at
        'that intersect the river, at the start of the day

    StartDepStorOffRiverM3 As Double 'The water stored in intact depressions,
        'that do not intersect the river, at the start of day

    StartSoilCapM As Double 'The soil capacity of the watershed'at the start
        'of the day.  When soil capacity is higher, the soil is dryer.

    InflowM3 As Double 'The inflow into the subwatershed from upstream
        'subwatersheds
```

```
    OutflowM3 As Double 'The outflow from the subwatershed to downstream
        'subwatersheds

End Type

'The following Type is used to hold the gage information that is used
'in the model.  The external array GageArray() is dimensioned with two
'indices, the 1st being the gage number assigned by the program (which is
'the VBGageID refered to the WS() array e.g., WS(5).VBGageID is the gage ID
'of the 5th subwatershed.)
'The second index is the date offset from the start date (e.g. this index
' = 0 a the start date.  The ReDim declaration occurs in GetPrecipGages
'The 1st two variables, SnoPlusRain and TAvgF are read from a table
'for the gage in the Subroutine GetPrecipGages.
'The 2nd two variables, MovingAverageF and Regime, are
'calculated in the subroutine SetSeasonBreaks--see comments in that
'subroutine for clarification.
Private Type GageType
    SnoPlusRainM As Single 'The amount of precipitation + snowmelt
    TAvgF As Single 'The day's average temperature
    MovingAverageF As Single 'The moving average temperature calculated
        'in the subroutine SetSeasonBreaks
    Regime As Integer 'The regime number calculated in SetSeasonBreaks
End Type

'The following Type is used to the hold the Regime information for model.
'A regime provides seasonal and geographically based parameters; so that
'for any particular day for a particular subwatershed, a certain Regime
'applies.  These parameters are read from the REGIMES table of the model.
'The external array RegimeRegionArray() will be dimensioned with this
'type and with two indices.  The first index will be the Regime
'(either 1 for low infilration/winter or 2 for high infiltation/summer)
'This regime (1 or 2) is deterimined by every date of the simulation for
'each Precipation Gage in the model (in SetSeasonBreaks)
''and the second index is the (geographic) region number assigned to each
'subwatershed, found in the WS() array for each subwatershed.  Since each
'subwatershed is also assigned a particular precipatation gage, and each
'precipitation gage is assigned to a particular regime based on the
'date, the Regime
'(1 or 2) and the region are unique for each subwatershed on particular day
'of the simulation.
'For example, RegimeRegionArray(2, 3).PercMPD would
'would contain the maximum percolation for the high infiltration (2) regime of the #3
Region
'of the model.
Private Type RegimeType
    EntryThresholdF As Double 'The threshold temperature at which
        'the regimes switch.  This is based on the moving average
        'temperature and is explained in the subroutine SetSeasonBreaks

    InfilMPD As Double 'The potential infiltration if the soil is dry

    PercMPD As Double 'The potential percolation if the soil is dry

    ETCoef As Double 'The evapotranspiration coefficient; used to
        'determine evaportranspiration in conjunction with the current soil
        'saturation level

    UpperZoneM As Double 'The length of the upper zone.  Percolation can
        'only take place with the SoilCapacity is less than the upper zone
        'distrance

    PotentialSoilCapM As Double 'The potential soil capacity, or how much
        'water the soil can hold.

    InitialSoilCapM As Double 'The soil capacity on the 1st day of the
        'simulation.  Higher soil capacity indicates dryer soil.
```

```
    NextDayFractionalFlow As Double 'A parameter which determines how much
        'of the subbasins outflow is shifted to the following simulation day's
        'inflows to it's downstream subbasin.  A higher causes greater
        'flow attenuation.

    TodaysFractionalFlow As Double 'This is not read from the REGIMES
        'table but calculated as = 1 - NextDayFractionalFlow. It is the
        'fraction of outflow which appears in the downstream subbasin on
        'the same simulation day.

End Type

'The following type is for any (optional) lake data in the model.
'The array LakeTable is dimensioned with this type, using two
'indices.  All the lake data is read into this one array.
''The 1st is the lake number.  The 2nd index is the row
'number of the that lake's table -- the table for each lakes
'is composed of various rows.  For example LakeTable(3, 8).VolumeM3
'would the the Volume of the the 8th row from the 3rd lake's table.
Private Type LakeTableType
    VolumeM3 As Double 'Storage volume
    AreaM2 As Double 'surface area at the storage volume above
    FlowM3PD As Double 'flow downstream in cubic meters per day at
        'this storage volume

End Type

Dim WS() As WaterShedType 'WS() Will be ReDim'd as 1 dim array
    'where the index is the Subbasin sequence number (SubBSeq)
    'from the model's WSHEDS table, as explained before in the
    'Private Type WaterShedType declaration.

Dim ST() As SimulTraceType 'ST() Will be ReDim'd as 2 dim array
    'as explained before in the
    'Private Type SimulTraceType declaration

Dim GageArray() As GageType 'GageArray() Will be ReDim'd as 2 dim array
    'as explained earlier in the
    'Private Type GageType declaration

Dim GageList() As String 'Has the names of the gages.  These are set in
    'GetPrecipGages.  This array is ReDim'd in GetPrecipGages.  The index
    'is program's internal Gage ID assigned in GetPrecipGages.

Dim LakeTable() As LakeTableType 'Contains the lake tables as
    'described in the Private Type LakeTableType declaration

Dim NumPrecipGages As Integer 'The total number of precip gages used for the simulation

Dim EvapArrayM() As Single ' Stores the daily evaporation in meters
    'With the array dimension the date offset = CurrentDate - StartDate
Dim InputStream() As Single 'Stores the Cubic Meters per day of flow of each input stream.
    ' 1st array dim is InputStream ID, 2nd Array dim is\
    'the date offset = CurrentDate - StartDate
    'a typical reference might look like: InputStream(StreamID, CurrentDate - StartDate)

Dim RegimeRegionArray() As RegimeType 'Will be ReDim'd as explained
    'earlier in the Private Type RegimeType declaration

Dim NumRegions As Long 'The number of (geographic) regions.  This is
    'set in ReadRegimeParams

Dim LongestLakeTableSize As Integer 'The greatest number of rows
    'used by any lake in the lake table; used to ReDim the LakeTable()
    'array
```

```
Dim STArrayDim1 As Long '1st dimension of ST() SimulTraceType Array
    'see comments on the Private Type SimulTraceType declaration

Dim ModelWshedTable As String 'The table name of the model's watershed
    'table.  This the table for the restored watersheds.
    'This is assigned in SetModelTableNames.

Dim ModelBaseCaseWShedTable As String 'The table name of the model's
    'unrestored watershed table. This is assigned in SetModelTableNames

Dim ModelRiverTable As String 'The table name of the model's river table.
    ' This is assigned in SetModelTableNames.  This table is used during
    'model generation, not during the simulation runs.

Dim ModelRegimeTable As String 'The table name of the model's regime table.
    'This is assigned in SetModelTableNames.  This table contains all the
    'parameters to use, by Regime and Region.

Dim StartDate As Date 'The simulation start date, which is read from
    'The launch screen, or set to a default value.  It is set in
    'GetSimulLength

Dim EndDate As Date 'Simulation end date.  Set in GetSimulLength.

Dim RestorationTransitionDate As Date 'That date at which to convert to
    'the restored Watershed information.
    'This is set in InitializeExternalVariables

Dim SimulDays As Long ' number of days the simulation will last, equal
    'to EndDate - StartDate

Dim NumSubBasins As Long 'The number of subbasins, from the watershed
    'table of the model.

'The following four variables are set in ReadInWaterSheds, and
'are based on summary totals of information from the model's watershed
'table.

Dim TotalIntactDepStorOnRiverM3 As Double 'The total volume of intact
    'depressional storage which intersects the river.

Dim TotalIntactDepStorOffRiverM3 As Double

Dim TotalNonDepAreaOnRiverM2 As Double
Dim TotalNonDepAreaOffRiverM2 As Double

Dim TotalOutletOutflowM3 As Double

Dim ExtraInflowTable() As String
Dim rstExtraInflowTable() As DAO.Recordset

Dim ExtraOutflowTable() As String
Dim rstExtraOutflowTable() As DAO.Recordset

Dim NumExtraInflowTables As Integer
Dim NumExtraOutflowTables As Integer

Option Compare Database 'Sets sort order for string comparisons; not used here

Private Sub CreateBoxModelInput_Click()

    Dim dbs As DAO.Database
    Dim rstColumnList As DAO.Recordset
    Dim rstRest As DAO.Recordset
    Dim rstSimul As DAO.Recordset
    Dim rstBoxModel As DAO.Recordset
    Dim qdf As DAO.QueryDef
```

```
    Dim fldLoop As DAO.Field
    Dim strSQL As String
    Dim strUnion As String
    Dim ExtractionTable As String
    Dim FirstRecord As Boolean
    Dim QueryName1() As String, QueryName2() As String
    Dim NumOutputColumns As Integer, ColumnNumber As Integer
    Dim C As String, WriteString As String

    Set dbs = CurrentDb

'    DeleteTableWithoutWarning ("BoxModelInpTable")

'    strSQL = "CREATE TABLE BoxModelInpTable " & _
'    "([Year] integer, " & _
'    "[Month] integer, " & _
'    "[ColumnName] char(15), " & _
'    "[Simul] char(3), " & _
'    "[RestorationLevel] char(1), " & _
'    "[AcreFt] single, " & _
'    "PRIMARY KEY ([Year], [Month], [ColumnName], [Simul], [RestorationLevel]));"
'    Set qdf = dbs.CreateQueryDef("", strSQL)
'    qdf.Execute
'    dbs.TableDefs.Refresh

    strSQL = "DELETE * FROM BoxModelInpTable;"
    Set qdf = dbs.CreateQueryDef("", strSQL)
    qdf.Execute
    dbs.TableDefs.Refresh


    strSQL = "SELECT * FROM BoxModelOutputColumns ORDER BY ColumnNo;"
    Set rstColumnList = dbs.OpenRecordset(strSQL)
    rstColumnList.MoveLast
    NumOutputColumns = rstColumnList.RecordCount
    rstColumnList.MoveFirst

    ReDim QueryName1(NumOutputColumns) As String
    ReDim QueryName2(NumOutputColumns) As String

    ColumnNumber = 1
    Do Until rstColumnList.EOF
        strSQL = "SELECT Simul FROM SimulList GROUP BY Simul ORDER BY Simul"
        Set rstSimul = dbs.OpenRecordset(strSQL)

        strSQL = "SELECT RestorationLevel FROM RestorationLevels GROUP BY RestorationLevel
ORDER BY RestorationLevel"
        Set rstRest = dbs.OpenRecordset(strSQL)

        rstSimul.MoveFirst
        FirstRecord = True
        Do Until rstSimul.EOF
            If rstSimul![Simul] <> "COMP" Then
                rstRest.MoveFirst
                If rstSimul![Simul] = "006" Then
                    FirstRecord = True
                    strUnion = strUnion & ";"
                    QueryName1(ColumnNumber) = "TempBoxModelInput" &
rstColumnList![ColumnName] & "_Part1"
                    DeleteQueryWithoutWarning (QueryName1(ColumnNumber))
                    Set qdf = dbs.CreateQueryDef(QueryName1(ColumnNumber), strUnion)
                    dbs.QueryDefs.Refresh
                    strUnion = ""
                End If
                Do Until rstRest.EOF
'                    txtSimulation.Value = rstSimul![Simul]
'                    txtRestorationLevel.Value = rstRest![RestorationLevel]
```

```
                    ExtractionTable = rstColumnList![TablePrefix] & "_" & rstSimul![Simul]
& "_" & rstRest![RestorationLevel]
                    Me.Repaint
                    If FirstRecord = True Then
                        strUnion = "SELECT ALL "
                    Else
                        strUnion = strUnion & Chr$(13) + Chr$(10) & "UNION ALL SELECT ALL "
                    End If
                    strUnion = strUnion & "'" & rstColumnList![ColumnName] & "'" & " AS
ColumnName, "
                    strUnion = strUnion & "'" & rstSimul![Simul] & "'" & " AS Simul, " & _
                                "'" & rstRest![RestorationLevel] & "'" & " AS
RestorationLevel, "
                    strUnion = strUnion & "[Date], [CMPD] FROM " & ExtractionTable & " "
                    If rstSimul![Simul] = "001" And rstRest![RestorationLevel] = "A" Then
                        'for the 001/A simul gives all dates, for others only gives after
10/1/2000
                    Else
                        strUnion = strUnion & "WHERE [Date] >= #10/1/2000#"
                    End If
                    rstRest.MoveNext
                    FirstRecord = False
                Loop
            End If
            rstSimul.MoveNext
        Loop
        FirstRecord = True
        strUnion = strUnion & ";"
        QueryName2(ColumnNumber) = "TempBoxModelInput" & rstColumnList![ColumnName] &
"_Part2"
        DeleteQueryWithoutWarning (QueryName2(ColumnNumber))
        Set qdf = dbs.CreateQueryDef(QueryName2(ColumnNumber), strUnion)
        dbs.QueryDefs.Refresh

        StatusBox.Value = "Adding to box model input information from " &
rstColumnList![ColumnNo] & ". " & rstColumnList![ColumnName]
        Me.Repaint

        'Add the data from query #1 to BoxModelInpTable
        strSQL = "INSERT INTO BoxModelInpTable ( [Year], [Month], ColumnName, Simul,
RestorationLevel, AcreFt ) " & _
                "SELECT Year, Month, ColumnName, Simul, RestorationLevel,
Sum([CMPD])/1233.48 AS AcreFt " & _
                "FROM " & QueryName1(ColumnNumber) & " AS ModelOut INNER JOIN
DateListYearMonth ON ModelOut.Date = DateListYearMonth.Date " & _
                "GROUP BY Year, Month, ColumnName, Simul, RestorationLevel;"
        Debug.Print strSQL
        Set qdf = dbs.CreateQueryDef("", strSQL)
        qdf.Execute

        'Add the date from query #2 to BoxModelInpTable
        strSQL = "INSERT INTO BoxModelInpTable ( [Year], [Month], ColumnName, Simul,
RestorationLevel, AcreFt ) " & _
                "SELECT Year, Month, ColumnName, Simul, RestorationLevel,
Sum([CMPD])/1233.48 AS AcreFt " & _
                "FROM " & QueryName2(ColumnNumber) & " AS ModelOut INNER JOIN
DateListYearMonth ON ModelOut.Date = DateListYearMonth.Date " & _
                "GROUP BY Year, Month, ColumnName, Simul, RestorationLevel;"
        Set qdf = dbs.CreateQueryDef("", strSQL)
        qdf.Execute

        DeleteQueryWithoutWarning (QueryName1(ColumnNumber))
        DeleteQueryWithoutWarning (QueryName2(ColumnNumber))

        strUnion = ""

        rstColumnList.MoveNext
```

```
        ColumnNumber = ColumnNumber + 1
    Loop

    Debug.Print strUnion
    rstRest.Close
    rstSimul.Close

    strSQL = "SELECT * FROM ReportBoxModelInputXTab;"
    Set rstBoxModel = dbs.OpenRecordset(strSQL)
    rstBoxModel.MoveFirst

    Open "BOXMODELINPUT.TXT" For Output As #1

    Print #1, "#Devils Lake Upper Basin Simulated Flows in Acre-Feet"
    Print #1, "#WEST Consultants, Inc."
    Print #1, "#PRINET V1.0, Data extracted on " & Date & " " & Time
    Print #1, "#Restoration Level A: Existing Conditions, B: 25%, C:50%, D:75%, E:100%"
    Print #1, "#Restoration begins on 10/1/2002 for all simulations"
    Print #1, "#Only drained depressions >= 0.5 feet average depth candidates for
restoration"
    Print #1, "#Column headings in the next row"

    WriteString = "#"
    For Each fldLoop In rstBoxModel.Fields ' writes out the titles
        WriteString = WriteString & fldLoop.Name & " "
    Next fldLoop

    Print #1, WriteString

    Do Until rstBoxModel.EOF
        WriteString = ""
        For Each fldLoop In rstBoxModel.Fields
            If fldLoop.Name < "99" Then 'for the numeric value fields
                WriteString = WriteString & Format(rstBoxModel.Fields(fldLoop.Name), "0.#")
& " "
            Else 'for the row identifiers
                WriteString = WriteString + CStr(rstBoxModel.Fields(fldLoop.Name)) & " "
            End If
        Next fldLoop
        Print #1, WriteString
        rstBoxModel.MoveNext
    Loop

    Close #1


    dbs.Close
    MsgBox ("Done")


End Sub

Private Sub CreateRestoration_Click()

    Dim dbs As DAO.Database
    Dim rst As DAO.Recordset
    Dim qdf As DAO.QueryDef
    Dim tdf As DAO.TableDef
    Dim fld As DAO.Field
    Dim rstDepTable As DAO.Recordset
    Dim rstRest As DAO.Recordset
    Dim strSQL As String
    Dim ImpPrefix As String
    Dim ImpDepTableBase As String
    Dim ImpDepTable As String
    Dim TotalRestorableVolM3 As Double
    Dim TotalRestorableAreaM2 As Double
```

```
    Dim RunningTotalVolM3 As Double
    Dim RunningTotalAreaM2 As Double

    Set dbs = CurrentDb

    ImpPrefix = "IMP_" & txtWaterShed.Value & "_"
    ImpDepTableBase = ImpPrefix & "DEPTABLE" & "_A"

    'Calculate the restorable depressional volume and area
    strSQL = "SELECT Sum(VOLM3) AS SumOfVOLM3, Sum(AREA_M2) AS SumOfAREA_M2 " & _
            "FROM " & ImpDepTableBase & " " & _
            "WHERE ([DEPTYPE123]=2) AND ([DEPTH_M] >= 0.1524);"
    Set rst = dbs.OpenRecordset(strSQL)
    Debug.Print strSQL
    rst.MoveFirst
    TotalRestorableVolM3 = rst![SumOfVOLM3]
    TotalRestorableAreaM2 = rst![SumOfAREA_M2]
    rst.Close

    Set tdf = dbs.TableDefs(ImpDepTableBase)
    If Not IsField(tdf, "RandomNum") Then
        Set fld = tdf.CreateField("RandomNum", dbDouble)
        tdf.Fields.Append fld
    End If

    If Not IsField(tdf, "RunningTotalFracVol") Then
        Set fld = tdf.CreateField("RunningTotalFracVol", dbDouble)
        tdf.Fields.Append fld
    End If

    If Not IsField(tdf, "RunningTotalFracArea") Then
        Set fld = tdf.CreateField("RunningTotalFracArea", dbDouble)
        tdf.Fields.Append fld
    End If
    dbs.TableDefs.Refresh

    ' Open the "A" imported depression tables and clear all random numbers
    strSQL = "SELECT * FROM " & ImpDepTableBase & ";"
    Set rstDepTable = dbs.OpenRecordset(strSQL)
    rstDepTable.MoveFirst
    Do Until rstDepTable.EOF
        rstDepTable.Edit
        rstDepTable![RandomNum] = Null
        rstDepTable.Update
        rstDepTable.MoveNext
    Loop
    rstDepTable.Close

    ' Open the "A" imported depression tables and assign random numbers to the restoration
candidates
    strSQL = "SELECT * FROM " & ImpDepTableBase & " " & _
            "WHERE ([DEPTYPE123]=2) AND ([DEPTH_M] >= 0.1524);"
    Set rstDepTable = dbs.OpenRecordset(strSQL)
    rstDepTable.MoveFirst
    Do Until rstDepTable.EOF
        rstDepTable.Edit
        rstDepTable![RandomNum] = Rnd()
        rstDepTable.Update
        rstDepTable.MoveNext
    Loop
    rstDepTable.Close

    ' Extract the records of the imported depression table, selecting only candidates for
restoration, and sort by the random number
    ' The depressions will be restored in the order the appear in this query
    strSQL = "SELECT * FROM " & ImpDepTableBase & " " & _
            "WHERE(DEPTYPE123) = 2 And (DEPTH_M) >= 0.1524 ORDER BY RandomNum;"
```

```
    Set rstDepTable = dbs.OpenRecordset(strSQL)
    rstDepTable.MoveFirst

    RunningTotalVolM3 = 0#
    RunningTotalAreaM2 = 0#

    With rstDepTable
        Do Until .EOF
            RunningTotalVolM3 = RunningTotalVolM3 + ![VOLM3]
            RunningTotalAreaM2 = RunningTotalAreaM2 + ![AREA_M2]
            .Edit
            ![RunningTotalFracVol] = RunningTotalVolM3 / TotalRestorableVolM3
            ![RunningTotalFracArea] = RunningTotalAreaM2 / TotalRestorableAreaM2
            .Update
            .MoveNext
        Loop
    End With
    rstDepTable.Close


    strSQL = "SELECT * FROM RestorationLevels ORDER BY RestorationLevel"
    Set rstRest = dbs.OpenRecordset(strSQL)

    rstRest.MoveFirst
    Do Until rstRest.EOF
        If rstRest![RestorationLevel] <> "A" Then
            txtRestorationLevel.Value = rstRest![RestorationLevel]
            Me.Repaint
            ImpDepTable = ImpPrefix & "DEPTABLE" & "_" & txtRestorationLevel.Value
            DeleteTableWithoutWarning (ImpDepTable)
            strSQL = "SELECT POLYID, DEPTH_M, AREA_M2, DEPTYPE123, VOLM3, INTERRIV INTO " &
ImpDepTable & " " & _
                    "FROM " & ImpDepTableBase & ";"
            Set qdf = dbs.CreateQueryDef("", strSQL)
            qdf.Execute
            dbs.TableDefs.Refresh

            strSQL = "UPDATE " & ImpDepTableBase & " AS BASE INNER JOIN " & ImpDepTable & "
AS NEW ON " & _
                    "BASE.POLYID = NEW.POLYID SET NEW.DEPTYPE123 = 1 " & _
                    "WHERE (((NEW.DEPTYPE123)=2) AND ((BASE.RunningTotalFracVol)<= " &
rstRest![FractionRestorationVol] & "));"
            Debug.Print strSQL
            Set qdf = dbs.CreateQueryDef("", strSQL)
            qdf.Execute
            Call CreateModel_Click

            Call SetModelTableNames
            strSQL = "UPDATE " & ModelBaseCaseWShedTable & " AS BASE INNER JOIN " &
ModelWshedTable & " AS NEW ON BASE.SubBSeq = NEW.SubBSeq SET NEW.InitWaterM3 =
BASE.InitWaterM3, NEW.InputStreamTable = [BASE].[InputStreamTable], NEW.LakeID =
[BASE].[LakeID], NEW.ExtraInflowID = [BASE].[ExtraInflowID], NEW.ExtraOutflowID =
[BASE].[ExtraOutflowID];"
            Set qdf = dbs.CreateQueryDef("", strSQL)
            qdf.Execute

        End If
        rstRest.MoveNext
    Loop

    rstRest.Close
    dbs.Close

    txtRestorationLevel.Value = "A"
```

```
End Sub

Private Sub RunAllSimulAllModels_Click()
    ' Runs all future simulations for all models that are checked in the WaterShedList
table

Dim dbs As DAO.Database
Dim rstModel As DAO.Recordset
Dim strSQL As String

Set dbs = CurrentDb

strSQL = "SELECT * FROM WaterShedList ORDER BY ProcessOrder;"
Set rstModel = dbs.OpenRecordset(strSQL)
rstModel.MoveFirst

Do Until rstModel.EOF
    If rstModel![ToProcess] = True Then
        txtWaterShed.Value = rstModel![Watershed]
        Me.Repaint
        Call RunAllSimuls_Click
    End If
    rstModel.MoveNext
Loop

rstModel.Close
dbs.Close

End Sub

Private Sub RunAllSimuls_Click()

    Dim dbs As DAO.Database
    Dim rstRest As DAO.Recordset
    Dim rstSimul As DAO.Recordset
    Dim strSQL As String

    Set dbs = CurrentDb

    strSQL = "SELECT Simul FROM SimulList GROUP BY Simul ORDER BY Simul"
    Set rstSimul = dbs.OpenRecordset(strSQL)

    strSQL = "SELECT RestorationLevel FROM RestorationLevels GROUP BY RestorationLevel
ORDER BY RestorationLevel"
    Set rstRest = dbs.OpenRecordset(strSQL)

    rstSimul.MoveFirst
    Do Until rstSimul.EOF
        If rstSimul![Simul] <> "COMP" Then
            rstRest.MoveFirst
            Do Until rstRest.EOF
                txtSimulation.Value = rstSimul![Simul]
                txtRestorationLevel.Value = rstRest![RestorationLevel]
                Me.Repaint
                Call RunSimul_Click
                rstRest.MoveNext
            Loop
        End If
        rstSimul.MoveNext
    Loop

    rstRest.Close
    rstSimul.Close
    dbs.Close

End Sub
```

```
Private Sub RunSimul_Click()

Call InitializeExternalVariables 'Initializes external variables
Call SetModelTableNames 'gets the table names for the model
SimulDays = GetSimulLength 'sets the simulation length
NumSubBasins = ReadInWaterSheds(ModelBaseCaseWShedTable)
    'reads in the watersheds for the base case (unrestored) scenario

Call GetLakes 'Loads the lakes table into memory
Call GetPrecipGages 'Loads the needed precip gages into memeory
Call GetEvap 'loads the evaporation into memory
Call ReadRegimeParams 'reads in the regime tables
Call SetSeasonBreaks 'sets the seasons breaks
Call CreateTraceTable 'creates an empty SimulTrace table for debugging
If LaunchSimulation() = False Then 'Launches the simulation
    MsgBox ("Simulation Not Run")
End If
Call CreateOutputTable 'copies the simulation results to an output table
    'specific to the model, simulation run, and restoration level

'MsgBox ("Done")


End Sub

Private Function ReadInWaterSheds(WatershedTable As String)
    'reads in the watershed information found in the argument table
    ' WatershedTable

Dim dbs As DAO.Database
Dim rst As DAO.Recordset 'used to identify the watershed table's recordset
Dim rstCountInputStreams As DAO.Recordset 'used to identify the recordset used
    'to count the number of input streams in the model

Dim rstInputStream As DAO.Recordset 'used to identify input stream recordsets
Dim I As Long 'Index number for WS() array; represents SubBSeq, the subbasin
    'sequence number

Dim strSQL As String 'used to create SQL queries
Dim strSQLInputStream As String 'SQL statement to retrieve input streams
Dim TableName As String 'used to hold table names for use in queries
Dim NumInputStreams As Long 'The number of input streams into the model
Dim CurrentInputStream As Long 'The current input stream

Dim qdf As DAO.QueryDef

NumInputStreams = 0
CurrentInputStream = 1

StatusBox.Value = "Reading In Watersheds"
Me.Repaint

Set dbs = CurrentDb

' A query which extracts only the input streams into the watershed
strSQL = "SELECT InputStreamTable FROM " & WatershedTable & " WHERE (InputStreamTable) Is
Not Null ORDER BY InputStreamTable;"
Set rstCountInputStreams = dbs.OpenRecordset(strSQL)
If rstCountInputStreams.EOF Then 'if there are no input streams
    NumInputStreams = 0 'set number of input streams to zero
Else 'otherwise move to end of table and count how many there are
    rstCountInputStreams.MoveLast
    NumInputStreams = rstCountInputStreams.RecordCount
End If
rstCountInputStreams.Close

ReDim InputStream(NumInputStreams, SimulDays) As Single
```

```
'The following query extracts the entire watershed table for the model
strSQL = "SELECT * FROM " & WatershedTable & " ORDER BY SubBSeq;"
Set rst = dbs.OpenRecordset(strSQL)

rst.MoveLast
NumSubBasins = rst.RecordCount 'set the number of subbasins based on
    'how many records were found in the model's watershed table

If NumSubBasins = 0 Then
    MsgBox ("No Watersheds found in table")
    Resume Exit_ReadInWaterSheds
End If

' The following set the total intact depressional storage on and
' off river equal to zero
TotalIntactDepStorOnRiverM3 = 0#
TotalIntactDepStorOffRiverM3 = 0#

' The following set the total non depressional surface areas to zero
'(e.g. soil; consists of all non-depressional surface area plus
' the surface are of drained depressions
TotalNonDepAreaOnRiverM2 = 0#
TotalNonDepAreaOffRiverM2 = 0#

' The following dimensions the WS() array to hold the information from
' each of the subwatersheds of the model
ReDim WS(NumSubBasins) As WaterShedType

' Read in the model's WSHED Table
rst.MoveFirst
I = 1
Do Until rst.EOF
    WS(I).DepAreaOnRiverM2 = rst![DepAreaOnRiverM2]
    WS(I).DepVolOnRiverM3 = rst![DepVolOnRiverM3]
    WS(I).DepAreaOffRiverM2 = rst![DepAreaOffRiverM2]
    WS(I).DepVolOffRiverM3 = rst![DepVolOffRiverM3]

    WS(I).DSSubBSeq = rst![DSSubBSeq]
    WS(I).InitWaterM3 = rst![InitWaterM3]
    If Not IsNull(rst![InputStreamTable]) Then
        WS(I).InputStreamTable = rst![InputStreamTable]
    Else 'sets string equal to empty if there is no input stream
        WS(I).InputStreamTable = ""
    End If
    WS(I).PrecipGage = rst![PrecipGage]
    WS(I).Region = rst![Region]
    WS(I).WSAreaNonDepOnRiverM2 = rst![WSAreaNonDepOnRiverM2]
    WS(I).WSAreaNonDepOffRiverM2 = rst![WSAreaNonDepOffRiverM2]
    If Not IsNull(rst![ExtraInflowID]) Then 'these are inflow tables to be written out for
the model
        'at a particular subwatershed; ExtraInflowID is the table name
        WS(I).ExtraInflowID = rst![ExtraInflowID]
    End If
    If (Not IsNull(rst![ExtraInflowID])) Then 'if there is an output table
        NumExtraInflowTables = NumExtraInflowTables + 1 'increment output table count
        ExtraInflowTable(NumExtraInflowTables) = GetInflowTableName(txtWaterShed.Value &
"_" & rst![ExtraInflowID])
            'set the output table name

        DeleteTableWithoutWarning (ExtraInflowTable(NumExtraInflowTables))
            'deletes the table if it already exits

        ' creates a SQL make-table to query to create the table
        strSQL = "CREATE TABLE " & ExtraInflowTable(NumExtraInflowTables) & " " & _
        "([Date] date, " & _
        "[CMPD] single, " & _
```

```
        "PRIMARY KEY ([Date]));"
        Set qdf = dbs.CreateQueryDef("", strSQL)
        qdf.Execute
        dbs.TableDefs.Refresh
        Set rstExtraInflowTable(NumExtraInflowTables) =
dbs.OpenRecordset(ExtraInflowTable(NumExtraInflowTables))
            ' assigns the output table handle to the rstExtraInflowTable array

        WS(I).ExtraInflowIndex = NumExtraInflowTables
            'assigns an index number to the output table; this is used to retrieve
            'the just defined table handle

    Else 'there is no output table for this subwatershed
        WS(I).ExtraInflowIndex = 0
    End If

    If Not IsNull(rst![ExtraOutflowID]) Then 'these are outflow output tables for the model
        'at a particular subwatershed; ExtraOutflowID is the table name
        WS(I).ExtraOutflowID = rst![ExtraOutflowID]
    End If

    If (Not IsNull(rst![ExtraOutflowID])) Then 'if there is an outflow table
        NumExtraOutflowTables = NumExtraOutflowTables + 1 'increment outflow output table
count
        ExtraOutflowTable(NumExtraOutflowTables) = GetOutflowTableName(txtWaterShed.Value &
"_" & rst![ExtraOutflowID])
            'set the output table name

        DeleteTableWithoutWarning (ExtraOutflowTable(NumExtraOutflowTables))
            'deletes the table if it already exits

        ' creates a SQL make-table to query to create the table
        strSQL = "CREATE TABLE " & ExtraOutflowTable(NumExtraOutflowTables) & " " & _
        "([Date] date, " & _
        "[CMPD] single, " & _
        "PRIMARY KEY ([Date]));"
        Set qdf = dbs.CreateQueryDef("", strSQL)
        qdf.Execute
        dbs.TableDefs.Refresh
        Set rstExtraOutflowTable(NumExtraOutflowTables) =
dbs.OpenRecordset(ExtraOutflowTable(NumExtraOutflowTables))
            ' assigns the outflow output table handle to the rstExtraInflowTable array

        WS(I).ExtraOutflowIndex = NumExtraOutflowTables
            'assigns an index number to the output table; this is used to retrieve
            'the just defined table handle

    Else 'there is no output table for this subwatershed
        WS(I).ExtraOutflowIndex = 0
    End If

    If (Not IsNull(rst![LakeID])) Then 'there is a lake associated with this subwatershed
        WS(I).LakeID = rst![LakeID] 'assign the Lake ID
        WS(I).DepVolOnRiverM3 = 0# 'set to zero because overflow is based on table lookup
    Else 'there is no lake assigned to this subwatershed
        WS(I).LakeID = 0
    End If

    WS(I).SubBSeq = rst![SubBSeq]

    If I <> rst![SubBSeq] Then 'verifies that the subbasin sequence numbers
            'start with 1 and are in sequential order

        MsgBox ("Watershed Seq# " & rst![SubBSeq] & " is out of sequence")
    End If

    'The following update the Totals of Depressional Storage and non-depressional area
```

```
    TotalIntactDepStorOnRiverM3 = TotalIntactDepStorOnRiverM3 + rst![DepVolOnRiverM3]
    TotalIntactDepStorOffRiverM3 = TotalIntactDepStorOffRiverM3 + rst![DepVolOffRiverM3]
    TotalNonDepAreaOnRiverM2 = TotalNonDepAreaOnRiverM2 + WS(I).WSAreaNonDepOnRiverM2
    TotalNonDepAreaOffRiverM2 = TotalNonDepAreaOffRiverM2 + WS(I).WSAreaNonDepOffRiverM2

    ' read in the input stream, if any, from the appropriate table
    If WS(I).InputStreamTable = "" Then
        WS(I).VBInputStreamID = 0
    Else
        TableName = GetOutputTableName(rst![InputStreamTable])

        ' The following query extracts the input stream information
        strSQLInputStream = "SELECT [Date], [CMPD] FROM " & TableName & _
                " WHERE ((Date) >= #" & StartDate & "# And (Date) <= #" & EndDate & "#)
ORDER BY Date;"
        Set rstInputStream = dbs.OpenRecordset(strSQLInputStream)
        rstInputStream.MoveLast

        If rstInputStream.RecordCount < StartDate - EndDate + 1 Then
                'If true the extracted input stream record does have enought records
                'to span the complete set of simulation dates
            MsgBox ("The input stream table " & rst![InputStreamTable] & " does not have
enough records for the simulation dates.")
        End If

        rstInputStream.MoveFirst
        If rstInputStream![Date] <> StartDate Then
                'If true the extracted input stream record does not begin at the start date
of
                'the simulation
            MsgBox ("The input stream table " & rst![InputStreamTable] & " has an incorrect
start date for the start date of this simul")
        End If

        'Now need to read the input stream into memory and assign it a VBInputStreamID
        WS(I).VBInputStreamID = CurrentInputStream 'assigns an ID
        rstInputStream.MoveFirst
        Do Until rstInputStream.EOF
           InputStream(CurrentInputStream, CLng(rstInputStream![Date] - StartDate)) =
rstInputStream![CMPD]
                'copies the input stream into memory one day at a time

           rstInputStream.MoveNext
        Loop
        CurrentInputStream = CurrentInputStream + 1

        rstInputStream.Close

    End If

    rst.MoveNext
I = I + 1
Loop

'Calculate each Watershed's Non Depressional Area as a fraction of the total

For I = 1 To NumSubBasins 'updates the percentage of watershed area that is
        'non-depressional (e.g. not intact depressions) as a fraction of the
        'total watershed area

    WS(I).WSAreaNonDepFractionOfTotal = (WS(I).WSAreaNonDepOnRiverM2 +
WS(I).WSAreaNonDepOffRiverM2) / _
        (TotalNonDepAreaOnRiverM2 + TotalNonDepAreaOffRiverM2)

Next I

rst.Close
```

```
ReadInWaterSheds = NumSubBasins ' Returns the number of watersheds

Exit_ReadInWaterSheds:
    Exit Function

Err_ReadInWaterSheds:
    MsgBox Err.Description
    Resume Exit_ReadInWaterSheds

End Function
Private Function ReadInRestorationWaterSheds(RestorationWatershedTable As String)
    ' reads in the needed values for the restored watershed, other data
    ' from the table is presumed to be identical to the base case watershed

Dim dbs As DAO.Database
Dim rst As DAO.Recordset, rstCountInputStreams As DAO.Recordset, rstInputStream As
DAO.Recordset
Dim I As Long
Dim strSQL As String, strSQLInputStream As String
Dim TableName As String
Dim CurrentInputStream As Long
Dim qdf As DAO.QueryDef

'CurrentInputStream = 1

StatusBox.Value = "Reading In Restoration Watersheds"
Me.Repaint

Set dbs = CurrentDb

' Extracts the all records from the restored watershed table
strSQL = "SELECT * FROM " & RestorationWatershedTable & " ORDER BY SubBSeq;"
Set rst = dbs.OpenRecordset(strSQL)

' verifies the number of watersheds is the same as the base case model
rst.MoveLast
If rst.RecordCount <> NumSubBasins Then
    MsgBox ("The Restoration Watershed does not have the same number of watersheds as the
original.  Press cntrl-break and stop program.")
End If

' Initializes the totals
TotalIntactDepStorOnRiverM3 = 0#
TotalIntactDepStorOffRiverM3 = 0#
TotalNonDepAreaOnRiverM2 = 0#
TotalNonDepAreaOffRiverM2 = 0#

' Read in the Restoration WSHED values needed
rst.MoveFirst
I = 1
Do Until rst.EOF
    WS(I).DepAreaOnRiverM2 = rst![DepAreaOnRiverM2]
    WS(I).DepVolOnRiverM3 = rst![DepVolOnRiverM3]
    WS(I).DepAreaOffRiverM2 = rst![DepAreaOffRiverM2]
    WS(I).DepVolOffRiverM3 = rst![DepVolOffRiverM3]

    If WS(I).DSSubBSeq <> rst![DSSubBSeq] Then 'verifies that the downstream
        'sequence numbers for each watershed are identical with the base case model
        MsgBox ("There is a difference in watershed sequencing the in the restoration
watershed.  Hit Cntrl-break to stop.")
    End If


    If WS(I).PrecipGage <> rst![PrecipGage] Then 'verifies that the precip
            'gage is the same for each sub watershed
```

```
        MsgBox ("There is a difference in precip gage assignments in the restoration
watershed.  Hit cntrl-break to stop.")
    End If
    If WS(I).Region <> rst![Region] Then 'verifies that the region numbers
            'are the same for each subwatershed
        MsgBox ("There is a difference in region assignment in the restoration watershed.
Hit cntrl-break to stop.")
    End If

    WS(I).WSAreaNonDepOnRiverM2 = rst![WSAreaNonDepOnRiverM2]
    WS(I).WSAreaNonDepOffRiverM2 = rst![WSAreaNonDepOffRiverM2]

    If (Not IsNull(rst![LakeID])) Then 'verfies that LakeID assignements are the same as
base case
        If WS(I).LakeID <> rst![LakeID] Then
            MsgBox ("There is a difference in lake assigments in the restoration watershed.
Hit cntrl-break to stop.")
        End If
    End If

    ' verifies that the Subbasin sequence number is the same as base case model
    If WS(I).SubBSeq <> rst![SubBSeq] Then
        MsgBox ("The sub basin sequencing is different in the restoration watershed.  Hit
cntrl-break to stop.")
    End If

    If I <> rst![SubBSeq] Then 'verifies that sub basin numbers are sequential
        MsgBox ("Watershed Seq# " & rst![SubBSeq] & " is out of sequence")
    End If

    ' Updates totals
    TotalIntactDepStorOnRiverM3 = TotalIntactDepStorOnRiverM3 + rst![DepVolOnRiverM3]
    TotalIntactDepStorOffRiverM3 = TotalIntactDepStorOffRiverM3 + rst![DepVolOffRiverM3]

    TotalNonDepAreaOnRiverM2 = TotalNonDepAreaOnRiverM2 + WS(I).WSAreaNonDepOnRiverM2
    TotalNonDepAreaOffRiverM2 = TotalNonDepAreaOffRiverM2 + WS(I).WSAreaNonDepOffRiverM2

    rst.MoveNext
I = I + 1
Loop

'Calculate each Watershed's Non Depressional Area as a fraction of the total

For I = 1 To NumSubBasins

    WS(I).WSAreaNonDepFractionOfTotal = (WS(I).WSAreaNonDepOnRiverM2 +
WS(I).WSAreaNonDepOffRiverM2) / _
        (TotalNonDepAreaOnRiverM2 + TotalNonDepAreaOffRiverM2)

Next I

rst.Close
dbs.Close

End Function

Private Function Max(A As Variant, B As Variant)
    'calculates the maximum of 2 values

On Error GoTo Err_Max

If A > B Then
    Max = A
Else
    Max = B
End If
```

```
Exit_Max:
    Exit Function

Err_Max:
    MsgBox Err.Description
    Resume Exit_Max

End Function

Private Sub CreateTraceTable()
    ' Create the trace table used for debugging only
    ' The trace table is only used if the radio button for this
    ' selection is selected on the application's form

On Error GoTo Err_CreateTraceTable

Dim dbs As DAO.Database
Dim qdf As DAO.QueryDef

Dim strSQL As String

Set dbs = CurrentDb

dbs.TableDefs.Delete "SimulTrace"

' empties the template table
strSQL = "DELETE * FROM SimulTraceTemplate;"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute


' creates the SimulTrace table from the template
strSQL = "SELECT SimulTraceTemplate.* INTO SimulTrace " & _
    "FROM SimulTraceTemplate;"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute


dbs.Close

Exit_CreateTraceTable:
    Exit Sub

Err_CreateTraceTable:
    MsgBox Err.Description
    Resume Exit_CreateTraceTable

End Sub

Private Function LaunchSimulation()

' On Error GoTo Err_LaunchSimulation

Dim dbs As DAO.Database
Dim qdf As DAO.QueryDef
Dim CurrentDate As Date ' the current simulation date
Dim rstSimulTrace As DAO.Recordset ' to identify the trace table used for debugging
Dim rstOutput As DAO.Recordset ' to identify the output table for the model
Dim ArrayRow As Long 'Where in the 1st dim of ST() array the simulation is currently
located
Dim SubB As Long ' An identifer for the Sub basin sequence number used as an index
    'in the WS() array.

Dim Regime As Long ' The regime number (set to either 1 or 2)
Dim strSQL As String
Dim StartingGlobalWaterM3 As Double 'used to track total water in the system at the
    'beginning of the simulation -- for debugging only
```

```
Dim EndingGlobalWaterM3 As Double 'used to tract total water in the system at the
    'end of the simulation -- for debugging only

Dim DateZeroOffset 'A specifice of days from the current simulation date
    'all ST() records for this date are to be initialized

Dim TotalDepVolM3 As Double 'the total depressional volume a subwatershed

Dim I As Long

Set dbs = CurrentDb

' opens the trace table for output; this table is used for debugging
Set rstSimulTrace = dbs.OpenRecordset("SimulTrace")

' this query creates the SimulOutput table,
' the primary table that the program
' outputs the outlet flow as well as several other daily values to
strSQL = "CREATE TABLE SimulOutput " & _
    "([Date] date, " & _
    "[CMPD] single, " & _
    "[StorageVolUtil] single, " & _
    "[SoilCapacityFraction] single, " & _
    "[PrecipM3] single, " & _
    "[LostToETM3] single, " & _
    "[LostToEvapM3] single, " & _
    "[LostToPercM3] single, " & _
    "PRIMARY KEY ([Date]));"

' Deletes the SimulOutputTable if it already exists
If DoesObjectExist("Tables", "SimulOutput") <> "" Then
    DoCmd.DeleteObject acTable, "SimulOutput"
End If

' run the create-table query
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute

' opens the SimulOutput table
Set rstOutput = dbs.OpenRecordset("SimulOutput")

DateZeroOffset = 2 'sets the offset number of days; this number of days
    'away will be zeroed out in the ST() array in preparation to receive
    ' new data
STArrayDim1 = DateZeroOffset + 1 ' The dimension of the ST() array
ReDim ST(STArrayDim1 - 1, NumSubBasins) 'Dimensions the ST() array based
    ' the required number of days that need to be held in memory
    ' (the 1st index) and the number of subbasins (2nd index)

'Set up first Period Initial Values
ArrayRow = CLng(StartDate) Mod STArrayDim1 'This sets which 1st indext to use
    'in the ST() array; it cycles through the array based on the current date

For SubB = 1 To NumSubBasins 'Initializes the ST array for the 1st period
    'of the model
    Regime = GageArray(WS(SubB).VBGageID, StartDate - StartDate).Regime
        'sets the regime based on the gage of the subwatershed and the
        'start date of the simulation

    TotalDepVolM3 = WS(SubB).DepVolOnRiverM3 + WS(SubB).DepVolOffRiverM3
        'the total depressional volume of the watershed; used in the following
        ' if ... then statement

    If (TotalDepVolM3) > 0# Then 'there is water in the depresions
          'the following proportions out any initial water
          'between the on-river and off-river depressions based on their
```

```
            'storage capacities

        ST(ArrayRow, SubB).StartDepStorOnRiverM3 = _
            WS(SubB).InitWaterM3 * WS(SubB).DepVolOnRiverM3 / TotalDepVolM3
        ST(ArrayRow, SubB).StartDepStorOffRiverM3 = _
            WS(SubB).InitWaterM3 * WS(SubB).DepVolOffRiverM3 / TotalDepVolM3
    Else 'sets initial depressional water equal to zero
        ST(ArrayRow, SubB).StartDepStorOnRiverM3 = 0#
        ST(ArrayRow, SubB).StartDepStorOffRiverM3 = 0#
    End If

    ST(ArrayRow, SubB).StartSoilCapM = RegimeRegionArray(Regime,
WS(SubB).Region).InitialSoilCapM
        'sets initial soil capacity based on the regime (of the simulation start date)
        'and the region the subwatershed is assigned to

    StartingGlobalWaterM3 = StartingGlobalWaterM3 + ST(ArrayRow,
SubB).StartDepStorOnRiverM3 + _
        ST(ArrayRow, SubB).StartDepStorOffRiverM3 - ST(ArrayRow, SubB).StartSoilCapM * _
        (WS(SubB).WSAreaNonDepOnRiverM2 + WS(SubB).WSAreaNonDepOffRiverM2)
Next SubB


For CurrentDate = StartDate To EndDate
    StatusBox.Value = "Simulation Date = " & CurrentDate
    Me.Repaint
    If CurrentDate = RestorationTransitionDate Then
        Call ReadInRestorationWaterSheds(ModelWshedTable)
    End If
    ArrayRow = CLng(CurrentDate) Mod STArrayDim1 'sets which array row of the
        'ST() array to use based on the current date

        'The following line prepares the ArrayRow that is DateZeroOffset out for data entry
    Call InitializeArrayRow((ArrayRow + DateZeroOffset) Mod STArrayDim1, DateAdd("d",
DateZeroOffset, CurrentDate))
    For SubB = 0 To NumSubBasins 'Places date values in the ST array for
            'each subbasin
         ST(ArrayRow, SubB).SimulDate = CurrentDate
    Next SubB
    Call CalcOneDayFlow(ArrayRow, CurrentDate, rstSimulTrace, rstOutput)
        ' calls the routine that does the day's calculations

Next CurrentDate

' Closes all extra output tables for the model
For I = 1 To NumExtraInflowTables
    rstExtraInflowTable(I).Close
Next I

For I = 1 To NumExtraOutflowTables
    rstExtraOutflowTable(I).Close
Next I


rstSimulTrace.Close
rstOutput.Close
dbs.Close

ArrayRow = CLng(EndDate) Mod STArrayDim1
' determines how much total water is in the model at the end of run
' actually gives the water at the beginning of the last day of simulation
' used for debugging and water balance
For SubB = 1 To NumSubBasins
    EndingGlobalWaterM3 = EndingGlobalWaterM3 + ST(ArrayRow, SubB).StartDepStorOnRiverM3 _
        + ST(ArrayRow, SubB).StartDepStorOffRiverM3 - ST(ArrayRow, SubB).StartSoilCapM * _
        (WS(SubB).WSAreaNonDepOnRiverM2 + WS(SubB).WSAreaNonDepOffRiverM2)
Next SubB
```

```
' outputs the water balance; for debugging only
'MsgBox ("Starting Vol = " & StartingGlobalWaterM3 & " M3  and Ending Volume = " &
EndingGlobalWaterM3 & _
'                " Outflow = " & TotalOutletOutflowM3 & " M3   Net Increase = " & _
'                StartingGlobalWaterM3 - EndingGlobalWaterM3 - TotalOutletOutflowM3)

LaunchSimulation = True

Exit_LaunchSimulation:
    Exit Function

Err_LaunchSimulation:
    MsgBox Err.Description
    Resume Exit_LaunchSimulation

End Function

Private Sub InitializeArrayRow(RowNum As Long, RowDate As Date)
    ' Initializes the ST() array row to prepare it for data
    ' entry; these must be initialized because the simulation
    ' cycles through the ST() every few days

On Error GoTo Err_InitializeArrayRow

Dim SubBSeqCount As Long

For SubBSeqCount = LBound(ST, 2) To UBound(ST, 2)
    ST(RowNum, SubBSeqCount).SimulDate = RowDate
    ST(RowNum, SubBSeqCount).StartDepStorOnRiverM3 = 0#
    ST(RowNum, SubBSeqCount).StartDepStorOffRiverM3 = 0#
    ST(RowNum, SubBSeqCount).StartSoilCapM = 0#
    ST(RowNum, SubBSeqCount).InflowM3 = 0#
    ST(RowNum, SubBSeqCount).OutflowM3 = 0#
Next


Exit_InitializeArrayRow:
    Exit Sub

Err_InitializeArrayRow:
    MsgBox Err.Description
    Resume Exit_InitializeArrayRow

End Sub

Private Sub GetPrecipGages()

'On Error GoTo Err_GetPrecipGages

Dim dbs As DAO.Database
Dim rstNeededGages As DAO.Recordset 'A recordset that simply counts
    'the number of gages required based on query

Dim rstPrecipGages As DAO.Recordset 'A recordset with the records from
    'the PrecipGage table; a table which contains the cross-reference
    'of the gage names as refered to in the watershed table, and the actual
    'table names that contain the corresponding data

Dim rstGageData As DAO.Recordset 'Used to open the tables that have the gage
    'records and the Snowmelt / Rain calculations

Dim I As Long ' a counter
Dim G As Long ' a counter for the gage number
Dim D As Long ' a counter for the date offset
Dim strSQL As String
```

```
StatusBox.Value = "Reading Precip Gage Information"
Me.Repaint

Set dbs = CurrentDb

' The following query extracts a list of the needed gages
' from the model's watershed table
strSQL = "SELECT PrecipGage FROM " & ModelBaseCaseWShedTable & " " & _
        "GROUP BY PrecipGage ORDER BY PrecipGage;"
Set rstNeededGages = dbs.OpenRecordset(strSQL)

rstNeededGages.MoveLast
NumPrecipGages = rstNeededGages.RecordCount

' Dimensions the GageArray and GageList arrays
' GageArray contains the Snowmelt + Rain values for each gage for each day
' for the both the actual values (COMP) and for all the future simulations
' the GageList array contains the gage names
ReDim GageArray(1 To NumPrecipGages, SimulDays)
ReDim GageList(NumPrecipGages)

' The following extracts the entire PrecipGages table, the table that has
' the cross reference between the gage name and the tables that have the
' information for each gage
strSQL = "SELECT * FROM PrecipGages;"

Set rstPrecipGages = dbs.OpenRecordset(strSQL)

' Read in the Precip Gages
rstNeededGages.MoveFirst

G = 0
Do Until rstNeededGages.EOF
    G = G + 1 'sets the current gage index number
    GageList(G) = rstNeededGages![PrecipGage] 'assigns the name of the gage
        ' to the GageList() array

    ' locate the required gage in the PrecipGages table
    rstPrecipGages.FindFirst ("[PrecipGage] = '" & rstNeededGages![PrecipGage] & "'")


   If rstPrecipGages.NoMatch Then
        MsgBox ("Precip Gage " & rstNeededGages![PrecipGage] & " was not found.  Correct
ArcView tables and regenerate model.")
   End If



    ' depending on whether the simulation is a calibration run (COMP) or
    ' a simulation, sets up the query to extract the appropriate records
    If (txtSimulation.Value = "COMP") Then 'extracts only the observed values
    strSQL = "SELECT * " & _
        "FROM " & rstPrecipGages![TableName] & " " & _
        "WHERE [Sequence] = 'COMP' " & _
        "ORDER BY [SimulDate];"
    Else ' extracts the observed + the future simulation values for the simulation
            'requested.  There is no date overlap since the future simulation dates
            ' in the gage record do not overlap the observed ([Sequence] = COMP) records

            strSQL = "SELECT * " & _
        "FROM " & rstPrecipGages![TableName] & " " & _
        "WHERE ([Sequence] = 'COMP') OR ([Sequence] = '" & txtSimulation.Value & "') " & _
        "ORDER BY [SimulDate];"
    End If


    Debug.Print strSQL ' for debugging
```

```
    ' Extract the records based on the query text from the If ... Then statement
    Set rstGageData = dbs.OpenRecordset(strSQL)

    ' goes to the record corresponding to the start of the simulation
    rstGageData.FindFirst ("[SimulDate] = #" & StartDate & "#")

    D = 0 ' This is the number of days from the the StartDate of the simululation

    ' Move through the gage record extracted until reach the end or reach the
    ' end of the simulation
    Do Until rstGageData.EOF Or D > SimulDays - 1

        ' Verifies that the date of the gage record matches the expected value
        If D <> rstGageData![SimulDate] - StartDate Then
            MsgBox ("Discontinuity in Gage Data")
        End If

        ' Sets the gage date for that gage for the day
        GageArray(G, D).SnoPlusRainM = rstGageData![SnoPlusRain] * 0.0254
        GageArray(G, D).TAvgF = rstGageData![TAvg]
        rstGageData.MoveNext
        D = D + 1 'adds one to the day offset
    Loop
    If D < SimulDays - 1 Then ' reached the end of the record but not the end
        'of the simulation
        MsgBox ("Reached end of " & rstNeededGages![PrecipGages] _
                & "Gage record before End Date of " & EndDate)
    End If
    rstGageData.Close
    rstNeededGages.MoveNext 'goes to the next record which has the name
        'of the next required gage
Loop

rstNeededGages.Close
rstPrecipGages.Close
dbs.Close

For I = 1 To NumSubBasins 'cycles through all the subbasins
    For G = 1 To NumPrecipGages 'cycles through all the Precip gages
        If WS(I).PrecipGage = GageList(G) Then
            WS(I).VBGageID = G ' assigns the Gage ID value if the gage
                'for the subwatershed matches the gage name for that Gage
                'ID Value

        End If
    Next G
    If WS(I).VBGageID = 0 Then 'this should not occur, it indicates
            'an empty field in one of the PrecipGage fields in the watershed
            ' table

        MsgBox ("No gage ID was assigned to SubWS Seq # " & I)
    End If
Next I


Exit_GetPrecipGages:
    Exit Sub

Err_GetPrecipGages:
    MsgBox Err.Description
    Resume Exit_GetPrecipGages

End Sub


Private Function GetSimulLength()
```

```
    'obtains the start and end dates of the simulation

On Error GoTo Err_GetSimulLength

If txtSimulation.Value = "COMP" Then 'this is a calibration run
    If IsNull(SimulStartDate.Value) Then 'no start date was typed in
        SimulStartDate.Value = #10/1/1978# 'sets the default start date
    End If
    If IsNull(SimulEndDate.Value) Then 'no end date was typed in
        SimulEndDate.Value = #9/30/1999# 'sets the default end date
    End If
ElseIf txtSimulation.Value = "WET" Then 'this is the WET simulation
    SimulStartDate.Value = #10/1/1978# 'sets the appropriate dates
    SimulEndDate.Value = #9/30/2035#
Else 'this is neither a calibration run or the WET simulation
    SimulStartDate.Value = #10/1/1978# 'sets the appropriate dates
    SimulEndDate.Value = #9/30/2020#
End If

Me.Repaint 'refreshes the screen with start and end dates

StartDate = CDate(DateValue(SimulStartDate.Value))
EndDate = CDate(DateValue(SimulEndDate.Value))

' The function return variable is assigned to SimulDays at the Call source
GetSimulLength = EndDate - StartDate + 1 ' how many days the simulation will last

Exit_GetSimulLength:
    Exit Function

Err_GetSimulLength:
    MsgBox Err.Description
    Resume Exit_GetSimulLength

End Function

Private Sub GetEvap()
    ' gets the evaporation record

Dim dbs As DAO.Database
Dim rst As DAO.Recordset
Dim strSQL As String
Dim NumEvapRecords

StatusBox.Value = "Reading Evaporation Data"
Me.Repaint

Set dbs = CurrentDb

If (txtSimulation.Value = "COMP") Then 'this is a calibration run
    strSQL = "SELECT * " & _
        "FROM EvapSim " & _
        "WHERE [Sequence] = 'COMP' " & _
        "AND [SimulDate] >= #" & StartDate & "# AND [SimulDate] <= #" & EndDate & "# " & _
        "ORDER BY [SimulDate];"
Else 'this is a simulation, extracts the COMP ("observed") records as well as the
        'simulation values.  There should be no overlap in dates
    strSQL = "SELECT * " & _
        "FROM EvapSim " & _
        "WHERE ([Sequence] = 'COMP') OR ([Sequence] = '" & txtSimulation.Value & "') " & _
        "AND [SimulDate] >= #" & StartDate & "# AND [SimulDate] <= #" & EndDate & "# " & _
        "ORDER BY [SimulDate];"
End If

Set rst = dbs.OpenRecordset(strSQL)
rst.MoveLast
```

```
NumEvapRecords = rst.RecordCount 'sets how many evaporation records (e.g. dates) were
extracted

' Checks that number of evaporation records is enough to cover the period required
If NumEvapRecords <> EndDate - StartDate + 1 Then
    MsgBox ("The number of evap records indicates that it does not span the entire date
range required.")
End If

rst.MoveFirst 'moves to the 1st extracted record

' verifies that both the start date and end date of the
'simulation are in the evap record
rst.FindFirst ("[SimulDate] = #" & EndDate & "#")
If rst.NoMatch Then
    MsgBox ("End Date of " & EndDate & " not found in evap record")
End If
rst.FindFirst ("[SimulDate] = #" & StartDate & "#")
If rst.NoMatch Then
    MsgBox ("Start Date of " & StartDate & " not found in evap record")
End If


' Dimensions the EvapArrayM to the needed number of days
ReDim EvapArrayM(0 To (SimulDays - 1)) As Single

'Sets the evaporation for each day where the array index
' is the date offset
Do Until rst.EOF
    EvapArrayM(rst![SimulDate] - StartDate) = rst![EvapPropM]
    rst.MoveNext
Loop

rst.Close
dbs.Close

Exit_GetEvap:
    Exit Sub

Err_GetEvap:
    MsgBox Err.Description
    Resume Exit_GetEvap

End Sub

Private Function Min(A As Variant, B As Variant)
    ' Calculates the minimum of two values

On Error GoTo Err_Min

If A < B Then
    Min = A
Else
    Min = B
End If

Exit_Min:
    Exit Function

Err_Min:
    MsgBox Err.Description
    Resume Exit_Min

End Function

Private Sub ReadRegimeParams()
    ' reads in the regimes
```

```
Dim dbs As DAO.Database
Dim rstRegions As DAO.Recordset
Dim rstRegimes As DAO.Recordset
Dim Region As Long
Dim Regime As Long
Dim strSQL As String
Dim NumRegimes As Long 'Is set to the number of regimes
    'in each region as the Do loop cycles through the regions

StatusBox.Value = "Reading In Regimes"
Me.Repaint

Set dbs = CurrentDb

' Gets a list of Regions from the model's regime table
strSQL = "SELECT Region FROM " & ModelRegimeTable & "  GROUP BY Region ORDER BY Region;"
Set rstRegions = dbs.OpenRecordset(strSQL)
rstRegions.MoveLast
NumRegions = rstRegions.RecordCount
rstRegions.MoveFirst
If NumRegions = 0 Then
    MsgBox ("No Records in Regime Table.  Stop and fix.")
End If

' Dimensions the RegimeRegionArray to required dimensions
ReDim RegimeRegionArray(2, NumRegions)

For Region = 1 To NumRegions 'cycles through the extract regions of the model's Regime
table
    If (rstRegions![Region] <> Region) Then 'verifies the regions are in order
        MsgBox ("Regions are not in sequence.  Stop simululation and fix.")
    End If

    ' Extracts the records for the current region from the model's Regime table
    strSQL = "SELECT * FROM " & ModelRegimeTable & " WHERE (((Region) = " & Region &
"))ORDER BY Region, RegimeID;"
    Set rstRegimes = dbs.OpenRecordset(strSQL)

    rstRegimes.MoveLast
    NumRegimes = rstRegimes.RecordCount 'sets the number of records found for this Region
in
        'the model's Regime table

    rstRegimes.MoveFirst

    If NumRegimes <> 2 Then ' the number of Regimes should equal 2 for each Region
        MsgBox ("Number of Regimes for Region " & Region & " equals " & NumRegimes & ".  It
should = 2.  Stop and fix regimes table.")
    End If
    For Regime = 1 To NumRegimes 'cycles through the regimes
        With rstRegimes 'all record references are for this recordset
            If ![RegimeID] <> Regime Then 'verifies the Regimes are in order starting with
1
                MsgBox ("Regime sequence is incorrect - regime # must = record number in
Regime Table")
            End If

            'Sets all the Regime parameters for each region from the table
            RegimeRegionArray(Regime, Region).EntryThresholdF = ![EntryThresholdF]
            RegimeRegionArray(Regime, Region).ETCoef = ![ETCoef]
            RegimeRegionArray(Regime, Region).InfilMPD = ![InfilMPD]
            RegimeRegionArray(Regime, Region).NextDayFractionalFlow =
![NextDayFractionalFlow]
            RegimeRegionArray(Regime, Region).TodaysFractionalFlow = 1# -
RegimeRegionArray(Regime, Region).NextDayFractionalFlow
            RegimeRegionArray(Regime, Region).PercMPD = ![PercMPD]
```

```
            RegimeRegionArray(Regime, Region).PotentialSoilCapM = ![PotentialSoilCapM]
            RegimeRegionArray(Regime, Region).UpperZoneM = ![UpperZoneM]
            RegimeRegionArray(Regime, Region).InitialSoilCapM = ![InitialSoilCapM]
            .MoveNext
        End With 'ends the "With rstRegimes" statment
    Next Regime
    rstRegimes.Close
    rstRegions.MoveNext
Next Region

rstRegions.Close
dbs.Close



End Sub
Private Function SetSeasonBreaks()
    ' sets the Regimes (e.g. Seasons) based on the moving average of temperature
    ' for each gage, using the thresholds estableshied in the model's Regime table

Dim dbs As DAO.Database
Dim rst As DAO.Recordset, rstBD As DAO.Recordset
Dim qdf As DAO.QueryDef

Dim D As Date ' a temporary variable to hold dates
Dim Month As Long
Dim Day As Long
Dim Year As Long
Dim Index As Long ' used to hold the date offset, the number of days
    ' since the simulation start date
Dim G As Long ' used to cycle through the precipitation gages for this model
Dim Lag As Integer ' the Lag time in days
Dim K As Integer ' a counter
Dim Sum As Single 'used to sum the total degrees over the lag period
Dim Avg As Single 'used to hold the average temperature over the lag period
Dim Tl As Single 'The Regime = 1 entry threshold temperature
    'in degrees F.  When the moving
    'average over Lag days drops below this for the first time (from October -
    ' December), that and subsequent days are tagged as being in Regime 1.

Dim Th As Single ' The Regime = 2 entry threshold temperature
    'in degress F.  When the moving average of the previous Lag days goes
    'above this for the first time (in January - June), that and subsequent
    'days are tagged as being in Regime 2.

Dim Tcur As Single ' A holder for the current day's temperature at the gage
Dim strSQL As String
Dim Tp As String 'Indicates the current Regime.  If Tp = "H" then are currently
    'in Regime 2, if Tp = "L" then are currently in Regime 1.

Dim TableNameSeasons As String 'The name of the table to which to output the
    'Season/Regime information for this model.  This is an output table only and
    ' is not used by the program

'sets the table name to output the seasonal / Regime information to
TableNameSeasons = GetOutputTableName(txtWaterShed.Value & "_SEASONS")
DeleteTableWithoutWarning (TableNameSeasons) 'deletes table if it exists

Set dbs = CurrentDb

'Creates the season information output table
' This table has one record for each date in which the Regimes / Seasons
' changed for each gage.
strSQL = "CREATE TABLE " & TableNameSeasons & " " & _
     "([Gage] char(20), " & _
     "[StartDate] Date, " & _
     "[Season] char(10), " & _
```

```
     "[MovingAverage] double, " & _
     "PRIMARY KEY ([Gage], [StartDate]));"

Debug.Print strSQL 'for debugging

Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute
dbs.TableDefs.Refresh

' Deletes all records from SeasonBreakOutput, a table which
' records the daily regimes for each gage.  This table is used for
' queries that extract data from the simulation run.
strSQL = "DELETE * FROM SeasonBreakOutput;"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute

Set rst = dbs.OpenRecordset("SeasonBreakOutput")
Set rstBD = dbs.OpenRecordset(TableNameSeasons)

' The entry threshold to enter into Winter / Low infiltration is
' set -- it is only based on the values from Region 1
Tl = RegimeRegionArray(1, 1).EntryThresholdF

' The entry threshold to enter into Summer / High infiltration is
' set -- it is only based on the values from Region 1
Th = RegimeRegionArray(2, 1).EntryThresholdF

Lag = 30 'sets the lag time in days; moving averages of temperature will
    'be calculated back this many days

For G = 1 To NumPrecipGages 'cycles through each Precip Gage

    D = StartDate 'starts at the first date of the simulation
    Tp = " "
    Do While D <= EndDate 'loops until the end date of the simululation
        Index = CLng(D - StartDate) 'sets the date offset
        If (Index < Lag - 1) Then ' are not far enough along to take a moving
                'average

            Avg = 0
            If GageArray(G, 0).TAvgF > Th Then 'sets the initial Regime
                'based on whether the temperature on the 1st day of the run
                'is greate than the Th, the entry threshold for Regime 2.

                Tp = "H"
            Else
                Tp = "L"
            End If
            If D = StartDate Then 'writes out a record to the season output
                        'table for the start date of the run for this gage

                    rstBD.AddNew
                    rstBD![StartDate] = D
                    rstBD![Season] = Tp
                    rstBD![MovingAverage] = Avg
                    rstBD![Gage] = GageList(G)
                    rstBD.Update
            End If
        Else
            Month = CLng(Format(D, "mm")) 'extracts the month number from the date
            Sum = 0
            For K = 0 To Lag - 1 'adds up all the average temperatures over the lag
                    'period

                Sum = Sum + GageArray(G, Index - K).TAvgF
            Next K
            Avg = Sum / Lag 'calculates the moving average
```

```
            If ((Avg < Tl And Month >= 10 And Month <= 12) And Tp = "H") Then
                    'If this is true then were in Regime 2 (Tp = "H") and just
                    'transitioned into Regime 1 by the moving average temperature
                    ' (Avg) dropping below the threshold temperature Tl.

                ' Add a record to the season output table indicating the transition
                rstBD.AddNew
                rstBD![StartDate] = D
                rstBD![Season] = "L"
                rstBD![MovingAverage] = Avg
                rstBD![Gage] = GageList(G)
                rstBD.Update
                Tp = "L"
            ElseIf (Avg > Th And Month >= 1 And Month <= 6 And Tp = "L") Then
                    'If this is true then were in Regime 1 (Tp = "L") and just
                    'transitioned into Regime 2 by the moving average temperature
                    '(Avg) rising above the the threshold temperature Th.

                ' Add a record ot the season output table indicating the transition
                rstBD.AddNew
                rstBD![StartDate] = D
                rstBD![Season] = "H"
                rstBD![MovingAverage] = Avg
                rstBD![Gage] = GageList(G)
                rstBD.Update
                Tp = "H"
            End If
        End If

    If Tp = "H" Then 'Assign the regime for this gage and date offset (Index)
        GageArray(G, Index).Regime = 2
    Else
        GageArray(G, Index).Regime = 1
    End If
    GageArray(G, Index).MovingAverageF = Avg 'set the moving average in the gage
        'array

    ' Add a record to the SeasonBreakOutput table for this date and gage,
    ' indicating the regime.
    rst.AddNew
    rst![Season] = Tp
    rst![Date] = D
    rst![TAverage] = GageArray(G, Index).TAvgF
    rst![MovingAverage] = Avg
    rst![Gage] = GageList(G)
    rst.Update
    D = D + 1 'increments the current date by 1

    Loop

Next G

rst.Close
rstBD.Close
dbs.Close

Exit_SetSeasonBreaks:
    Exit Function

Err_SetSeasonBreaks:
    MsgBox Err.Description
    Resume Exit_SetSeasonBreaks

End Function

Function DoesObjectExist(ObjectType$, ObjectName$)
    'This function was copied from a Microsoft website
```

```
    'and was slightly modified

On Error Resume Next

Dim Found_Object, Find_Object As String, ObjectNum As Integer
Dim DB As Database, T As TableDef
Dim Q As QueryDef, C  As Container
Dim Msg As String
Found_Object = -1
Set DB = CurrentDb

Select Case ObjectType$
Case "Tables"

    Find_Object = DB.TableDefs(ObjectName$).Name

Case "Queries"

    Find_Object = DB.QueryDefs(ObjectName$).Name

Case Else

    If ObjectType$ = "Forms" Then
        ObjectNum = 1
    ElseIf ObjectType$ = "Modules" Then
        ObjectNum = 2
    ElseIf ObjectType$ = "Reports" Then
        ObjectNum = 4
    ElseIf ObjectType$ = "Macros" Then
        ObjectNum = 5
    Else
         Msg = "Object Name """ & ObjectType & """ is an invalid"
         Msg = Msg & " argument to function ObjectExists_20!"
         MsgBox Msg, 16, "ObjectExists_20"
         Exit Function

    End If

    Set C = DB.Containers(ObjectNum)
    Find_Object = C.Documents(ObjectName$).Name

End Select

If Err = 3265 Or Find_Object = "" Then

    Found_Object = ""

End If

DoesObjectExist = Found_Object

End Function

Private Sub CalcOneDayFlow(Row As Long, CurrentDate As Date, rst As DAO.Recordset,
rstOutput As DAO.Recordset)
    'Calculates one day of activity

Dim SubB As Long 'this will be the subbasin value in the for next loop
Dim PrecipM As Double 'This is read from the precip record for the gage and date
'The following variables are all set from the regime that the watershed pertains to for the
date in question
Dim PotentialInfilMPD As Double
Dim PotentialPercMPD As Double
Dim ETCoef As Double
Dim UpperZoneM As Double
Dim PotentialSoilCapM As Double
'The following are and/or intermediate values used in the this subroutine
```

```
Dim InfilM As Double
Dim PercM As Double
Dim SoilCapM As Double 'Soil capacity; greater soil capacity
Dim SoilEvapM As Double
Dim PrecipOnDepOnRiverM3 As Double 'Precipitation directly on depressions
    'that intersect the river network (on river)

Dim PrecipOnDepOffRiverM3 As Double 'Precipitation directly on depressions
    'that do not intersect the river network (off river)

Dim DepStorOnRiverM3 As Double 'Water storage of depressions on the river
Dim DepStorOffRiverM3 As Double 'Water storage of depressions off the river
Dim DepEvapOnRiverM3 As Double 'Evaporation of depressions on the river
Dim DepEvapOffRiverM3 As Double 'Evaporation of depressions off the river
Dim DepAreaForEvapOnRiverM2 As Double 'Surface are of on river depressions
    'when those depressions are full

Dim DepAreaForEvapOffRiverM2 As Double 'Surface are of off river depressions
    'when those depressions are full

Dim RunoffToDepStorOnRiverM3 As Double 'Runoff to on river depressions from
    'their contributing area in the same subbasin
    'not including the depressional area itself

Dim RunoffToDepStorOffRiverM3 As Double 'Runoff to off river depressions from
    'their contributing area in the same subbasin
    'not including the depressional area itself


Dim DSOutflowOnRiverM3 As Double 'The outflow from on river depressions to
    'the downstream subbasin (or outlet)  This outflow is caused by overtopping
    'except in the case of lakes which have a volume / outflow relationship

Dim OutflowOffRiverM3 As Double 'The outflow from off river depression to
    'the on river depression in the same subbasin.
    'This outflow is caused by overtopping

Dim Regime As Long 'The Regime (Season), 1 or 2; each Gage has a particular
    'Regime for each calendar date, and each subbasin is assigned to a gage.

Dim Region As Long 'The Region number as specified in the watersheds table,
    'the region is set by Subbasin

Dim DSSubB As Long 'The sequence number of the subbasin downstream of the
    'the current subbasin

Dim TotalWaterEndOfDayM3 As Double 'Total water in the system at the end of
    'the day; includes all subbasins.  Soil water is accounting as negative
    'soil capacity -- for debuggine purposes only

Dim TodaysDepEvapOnRiverM3 As Double 'Day's total evaporation from on river depressions
Dim TodaysDepEvapOffRiverM3 As Double 'Day's total evaporation from off river depressions
Dim TodaysETEvapM3 As Double 'Day's total volume of evapotranspirated water
Dim TodaysPercM3 As Double 'Day's total percolation volume of water
Dim TodaysPrecipM3 As Double 'Days total precipitation volume
Dim TodaysPercM As Double 'Days percolation distance; a weighted average
Dim TensionZoneCapFrac As Double 'Tension zone capacity fraction, set for
    'each subwatershed

Dim ETTensionZoneMultiplier As Double 'The multiplier which may reduce
    'ET due to soil dryness, set for each subwatershed

Dim RunoffM As Double 'Runoff in meters, set for each subwatershed
Dim SoilCapacityFraction As Double 'The fractional soil capacity, set
    'by each subwatershed

Dim MovingAverageF As Double 'The moving average temperate for the subbasin's
```

```
    'gage as determined in SetSeasonBreaks

Dim HighEntryThresholdF As Double 'The entry threshold to move into the 2
    'Regime (High infiltration) as set in the model's regime table; this is
    'used to in the final days of Regime 1 to slowly increase the potential
    'infiltatration to provide a smooth transition to Regime 2

Dim HighPotentialInfilMPD As Double 'The potential infiltration of the 2 Regime
    'this is used to in the final days of Regime 1 to slowly increase the potential
    'infiltatration to provide a smooth transition to Regime 2


'Initialize various variables
TotalWaterEndOfDayM3 = 0#
TodaysDepEvapOnRiverM3 = 0#
TodaysDepEvapOffRiverM3 = 0#
TodaysETEvapM3 = 0#
TodaysPercM3 = 0#
TodaysPrecipM3 = 0#
TodaysPercM = 0#
SoilCapacityFraction = 0#

For SubB = 1 To NumSubBasins 'cycle through the subbasins starting upstream

    If ST(Row, SubB).SimulDate <> CurrentDate Then 'for debugging
        MsgBox "Mismatch in Date in ST Array"
    End If

   'Write inflow to table if this watershed is tagged for output
   If WS(SubB).ExtraInflowIndex <> 0 Then
        rstExtraInflowTable(WS(SubB).ExtraInflowIndex).AddNew
        rstExtraInflowTable(WS(SubB).ExtraInflowIndex)![Date] = CurrentDate
        rstExtraInflowTable(WS(SubB).ExtraInflowIndex)![CMPD] = ST(Row, SubB).InflowM3
        rstExtraInflowTable(WS(SubB).ExtraInflowIndex).Update
    End If

    'The regime number is determined for this gage and the current date
    'Regimes are set seasonally according to the algorithm in the AssignRegimes fucntion
    'They are set seasonally for each gage.  The following equation looks up the gage
number for the current watershed
    'And finds what regime is associated with that gage for the current date
    Regime = GageArray(WS(SubB).VBGageID, CurrentDate - StartDate).Regime
    Region = WS(SubB).Region

    'All the regime variables are set
    PotentialInfilMPD = RegimeRegionArray(Regime, Region).InfilMPD 'Looks up the maximum
infiltration for the current regime
    If (Regime = 1) Then 'the following smooths the increase in at the end of winter; also
affects the beginning of winter
        HighPotentialInfilMPD = RegimeRegionArray(2, Region).InfilMPD
        MovingAverageF = GageArray(WS(SubB).VBGageID, CurrentDate -
StartDate).MovingAverageF
        HighEntryThresholdF = RegimeRegionArray(2, Region).EntryThresholdF
        If MovingAverageF < HighEntryThresholdF And MovingAverageF > 32# Then
            PotentialInfilMPD = Max(HighPotentialInfilMPD * ((MovingAverageF - 32#) /
(HighEntryThresholdF - 32#)) ^ 2#, PotentialInfilMPD)
        End If
    End If
    PotentialPercMPD = RegimeRegionArray(Regime, Region).PercMPD 'Looks up the maximum
percolation rate the current regime
    ETCoef = RegimeRegionArray(Regime, Region).ETCoef 'Looks up the EvapoTranspiration
coeffcient for the current regime
    UpperZoneM = RegimeRegionArray(Regime, Region).UpperZoneM 'Looks up the depth of the
upper zone for the current regime
    PotentialSoilCapM = RegimeRegionArray(Regime, Region).PotentialSoilCapM 'Looks up the
potential soil capacity for the current regime
```

```
    'Precipitation is determined
    PrecipM = GageArray(WS(SubB).VBGageID, CLng(CurrentDate - StartDate)).SnoPlusRainM

    'SoilCapacity is set to its starting value
    SoilCapM = ST(Row, SubB).StartSoilCapM 'sets initial soil capacity
    'Infiltration is determined; it is the minimum of Precipitation, Potential
Infiltration, and the Starting soil
    'capacity of the subbasin.
    'Adjust potential infiltration rate based on soil saturationm as in HMS Soil Moisture
Accounting
    PotentialInfilMPD = PotentialInfilMPD * SoilCapM / PotentialSoilCapM
    If PrecipM > 0# Then
        InfilM = Min(PrecipM, PotentialInfilMPD)
        InfilM = Min(InfilM, ST(Row, SubB).StartSoilCapM)
        'An intermediate soil capacity is calculated after infiltration but before
percolation
        SoilCapM = SoilCapM - InfilM 'starting capacity minus infiltration
    Else
        InfilM = 0# 'no infiltration takes place if no precip
    End If

    'Runoff from this subbasin's soil is calculated
    'it is the excess of precip over infiltration times non-depressional area) that goes
into it's own depressions is calculated
    If (PrecipM > InfilM) Then
        RunoffM = PrecipM - InfilM
        RunoffToDepStorOnRiverM3 = RunoffM * WS(SubB).WSAreaNonDepOnRiverM2
        RunoffToDepStorOffRiverM3 = RunoffM * WS(SubB).WSAreaNonDepOffRiverM2
    Else
        RunoffM = 0#
        RunoffToDepStorOnRiverM3 = 0#
        RunoffToDepStorOffRiverM3 = 0#
    End If

    'Percolation is determined.  SoilCapM recalculated to include percolation
    PotentialPercMPD = RegimeRegionArray(Regime, Region).PercMPD 'Looks up the maximum
percolation for the regime of the watershed
    If SoilCapM > UpperZoneM Then 'no percolation takes place because are not in the upper
zone
        PercM = 0# 'nothing happens
    ElseIf SoilCapM + PotentialPercMPD < UpperZoneM Then 'percolation would still leave
soil capacity in the upper zone
        SoilCapM = SoilCapM + PotentialPercMPD 'The Soil Capacity is increased by the
maximum percolation allowed
        PercM = PotentialPercMPD
    Else 'the only option left is that the starting soil capacity is in the upper zone, but
the maximum percolation would take it below the upper zone
        PercM = UpperZoneM - SoilCapM
        SoilCapM = UpperZoneM 'therefore only enough percolates for the Soil Capacity to
equal the upper zone limit
    End If
    'Add the subbasin's percolation volume to the day's total
    TodaysPercM3 = TodaysPercM3 + PercM * (WS(SubB).WSAreaNonDepOnRiverM2 +
WS(SubB).WSAreaNonDepOffRiverM2)

    'Precipitation on the depressional area is calculated - the full depressional area is
used
    If (PrecipM > 0#) Then
        PrecipOnDepOnRiverM3 = PrecipM * WS(SubB).DepAreaOnRiverM2
        PrecipOnDepOffRiverM3 = PrecipM * WS(SubB).DepAreaOffRiverM2
    Else 'no precipitation
        PrecipOnDepOnRiverM3 = 0#
        PrecipOnDepOffRiverM3 = 0#
    End If

    'Add the precipitation on the various portions of the subbasin
    'to the day's total precipitation
```

```
    TodaysPrecipM3 = TodaysPrecipM3 + _
        PrecipOnDepOnRiverM3 + PrecipOnDepOffRiverM3 + _
            (WS(SubB).WSAreaNonDepOnRiverM2 + WS(SubB).WSAreaNonDepOffRiverM2) * PrecipM

    'The precipitation is added to the depressional storage, along with starting value,
    'inflow from upstream, and runoff calculated above
    If (PrecipM > 0#) Then
        DepStorOnRiverM3 = ST(Row, SubB).StartDepStorOnRiverM3 + _
            ST(Row, SubB).InflowM3 + RunoffToDepStorOnRiverM3 + PrecipOnDepOnRiverM3
        DepStorOffRiverM3 = ST(Row, SubB).StartDepStorOffRiverM3 + _
            RunoffToDepStorOffRiverM3 + PrecipOnDepOffRiverM3
    Else 'no precipitation, just add upstream inflow
        DepStorOnRiverM3 = ST(Row, SubB).StartDepStorOnRiverM3 + ST(Row, SubB).InflowM3
        DepStorOffRiverM3 = ST(Row, SubB).StartDepStorOffRiverM3
    End If

    If WS(SubB).VBInputStreamID > 0 Then 'If there is an input stream this adds that inflow
        DepStorOnRiverM3 = DepStorOnRiverM3 + _
            InputStream(WS(SubB).VBInputStreamID, CLng(CurrentDate - StartDate))
        ST(Row, SubB).InflowM3 = ST(Row, SubB).InflowM3 +
InputStream(WS(SubB).VBInputStreamID, CLng(CurrentDate - StartDate))
    End If

    ' Then evap off the soil.  As water level drops soil capacity is increased
    ' Set TensionZoneCapFrac which is the fractional water capacity of the tension zone
(ten zone = Total Storage Potential - Upper Zone)
    SoilEvapM = -1# ' set this as negative for debugging purposes; if still negative later,
it wasn't set as it should have been
    If SoilCapM > UpperZoneM Then ' the upper zone is dry
        TensionZoneCapFrac = (SoilCapM - UpperZoneM) / (PotentialSoilCapM - UpperZoneM)
        If TensionZoneCapFrac >= 0.5 Then
            ETTensionZoneMultiplier = 1# - TensionZoneCapFrac
        ElseIf TensionZoneCapFrac <= 0.4 Then
            ETTensionZoneMultiplier = 1# 'here the Fractional Tension Zone Capacity is
between 0.4 and 0.5
        Else
            ETTensionZoneMultiplier = (0.5 - TensionZoneCapFrac) * 0.5 + 0.5
            If TensionZoneCapFrac < 0.4 Or TensionZoneCapFrac > 0.5 Then 'this is for
debugging purposes
                MsgBox ("Tension Zone Capacity out of range.  Stop and debug.")
            End If
        End If
        SoilEvapM = ETTensionZoneMultiplier * EvapArrayM(CurrentDate - StartDate) * ETCoef
    Else ' the upper zone has water and the tension zone is saturated -- the evaporation is
the full amount
        TensionZoneCapFrac = 0#
        ETTensionZoneMultiplier = 1#
        SoilEvapM = EvapArrayM(CurrentDate - StartDate) * ETCoef
    End If

    If SoilEvapM < 0# Then ' a check for debugging purposes
        MsgBox ("Soil Evap Not set. Debug Code")
    End If

    ' update the soil capacity based on the evaporation that was just calculated
    If SoilCapM + SoilEvapM > PotentialSoilCapM Then 'evaporation would take water level
below the soil capacity
        SoilEvapM = PotentialSoilCapM - SoilCapM 'set evaporation to just dry out the soil
        SoilCapM = PotentialSoilCapM 'reset the Soil Capacity to potential soil capacity
    Else 'the full evaporation takes place since there is enough soil capacity
        SoilCapM = SoilCapM + SoilEvapM
    End If

    If SoilEvapM < 0# Then 'for debugging purposes
        MsgBox ("Soil Evap is negative.  Stop program and debug.")
    End If
```

```
    TodaysETEvapM3 = TodaysETEvapM3 + SoilEvapM * _
        (WS(SubB).WSAreaNonDepOnRiverM2 + WS(SubB).WSAreaNonDepOffRiverM2)

    ' Find the surface area to evap off of in the ON river depressions
    If WS(SubB).DepVolOnRiverM3 = 0# Then 'if there is no depressional volume available
then evap. is zero
        DepAreaForEvapOnRiverM2 = 0#
    ElseIf DepStorOnRiverM3 = 0# Then 'if no water in depression then evap area is zero
        DepAreaForEvapOnRiverM2 = 0#
    ElseIf DepStorOnRiverM3 > WS(SubB).DepVolOnRiverM3 Then 'is already overtopped
        DepAreaForEvapOnRiverM2 = WS(SubB).DepAreaOnRiverM2
    Else 'This is the algorithm for setting the depressional area to evap off of
        DepAreaForEvapOnRiverM2 = Sqr(DepStorOnRiverM3 / WS(SubB).DepVolOnRiverM3) *
WS(SubB).DepAreaOnRiverM2
    End If

    If (WS(SubB).LakeID <> 0) Then 'There is a lake here; must override the depressional
area for evaporation
        DepAreaForEvapOnRiverM2 = LakeTable(WS(SubB).LakeID,
GetLakeTableRow(WS(SubB).LakeID, DepStorOnRiverM3)).AreaM2
        If DepAreaForEvapOnRiverM2 > WS(SubB).DepAreaOnRiverM2 Then 'If area exceeds area
given in watershed table
            DepAreaForEvapOnRiverM2 = WS(SubB).DepAreaOnRiverM2 'then area is set to
maximum regardless of what the lake table says
        End If
    End If

    ' Find the surface area to evap off of in the OFF river depressions
    If WS(SubB).DepVolOffRiverM3 = 0# Then 'if there is no depressional volume available
then evap. is zero
        DepAreaForEvapOffRiverM2 = 0#
    ElseIf DepStorOffRiverM3 = 0# Then 'if no water in depression then evap area is zero
        DepAreaForEvapOffRiverM2 = 0#
    ElseIf DepStorOffRiverM3 > WS(SubB).DepVolOffRiverM3 Then 'is already overtopped
        DepAreaForEvapOffRiverM2 = WS(SubB).DepAreaOffRiverM2
    Else 'This is the algorithm for setting the depressional area to evap off of
        DepAreaForEvapOffRiverM2 = Sqr(DepStorOffRiverM3 / WS(SubB).DepVolOffRiverM3) *
WS(SubB).DepAreaOffRiverM2
    End If

    'Evaporate the water from the depressions ON RIVER
    If DepAreaForEvapOnRiverM2 > 0# Then
        DepEvapOnRiverM3 = DepAreaForEvapOnRiverM2 * EvapArrayM(CurrentDate - StartDate)
        DepStorOnRiverM3 = DepStorOnRiverM3 - DepEvapOnRiverM3
        If DepEvapOnRiverM3 > DepStorOnRiverM3 Then 'everything in the depression
evaporates away
            DepEvapOnRiverM3 = DepStorOnRiverM3 'sets new evaporation volume
            DepStorOnRiverM3 = 0# 'zeros out the depressional storage
        End If
    Else
        DepEvapOnRiverM3 = 0#
    End If

    ' Evaporate the water from the depressions OFF RIVER
    If DepAreaForEvapOffRiverM2 > 0# Then
        DepEvapOffRiverM3 = DepAreaForEvapOffRiverM2 * EvapArrayM(CurrentDate - StartDate)
        DepStorOffRiverM3 = DepStorOffRiverM3 - DepEvapOffRiverM3
        If DepEvapOffRiverM3 > DepStorOffRiverM3 Then 'everything in the depression
evaporates away
            DepEvapOffRiverM3 = DepStorOffRiverM3 'sets new evaporation volume
            DepStorOffRiverM3 = 0# 'zeros out the depressional storage
        End If
    Else
        DepEvapOffRiverM3 = 0#
    End If

    'Increments the day's evaporation volume totals
```

```
    TodaysDepEvapOnRiverM3 = TodaysDepEvapOnRiverM3 + DepEvapOnRiverM3
    TodaysDepEvapOffRiverM3 = TodaysDepEvapOffRiverM3 + DepEvapOffRiverM3

    DSOutflowOnRiverM3 = 0#
    OutflowOffRiverM3 = 0#

    'Runoff from the off River depressions to the ON River Depressions
    'in the same subbasin if they are overflowing
    If DepStorOffRiverM3 > WS(SubB).DepVolOffRiverM3 Then 'the depression has overflowed
        OutflowOffRiverM3 = DepStorOffRiverM3 - WS(SubB).DepVolOffRiverM3
        OutflowOffRiverM3 = Min(OutflowOffRiverM3, DepStorOffRiverM3) 'flow cannot exceed
depressional storage
        DepStorOnRiverM3 = DepStorOnRiverM3 + OutflowOffRiverM3
        DepStorOffRiverM3 = WS(SubB).DepVolOffRiverM3  'Sets depressional storage equal to
the maximum
    Else
        OutflowOffRiverM3 = 0#
    End If

    'Initialize variable
    DSOutflowOnRiverM3 = 0#
    ' Then runoff from the on ON river depressions to downstream depressions if it is
overflowing
    If DepStorOnRiverM3 > WS(SubB).DepVolOnRiverM3 Then 'the depression has overflowed;
sent the water downstream
        DSSubB = WS(SubB).DSSubBSeq 'This is the identifier of the downstream subbasin
        DSOutflowOnRiverM3 = DepStorOnRiverM3 - WS(SubB).DepVolOnRiverM3
        If (WS(SubB).LakeID <> 0) Then 'There is a lake here; must reset flow based on
volume
            DSOutflowOnRiverM3 = LakeTable(WS(SubB).LakeID,
GetLakeTableRow(WS(SubB).LakeID, DepStorOnRiverM3)).FlowM3PD
        DSOutflowOnRiverM3 = Min(DSOutflowOnRiverM3, DepStorOnRiverM3) 'flow cannot exceed
depressional storage
        End If
        'The following adds the outflow from this Subbasin to the inflow from the
downstream subbasin
        'Send the amount of flow that appears downstream on the same day
        ST((Row) Mod STArrayDim1, DSSubB).InflowM3 = _
            ST((Row) Mod STArrayDim1, DSSubB).InflowM3 + DSOutflowOnRiverM3 *
RegimeRegionArray(Regime, Region).TodaysFractionalFlow
        'Send the amount of flow that appears downstram on the following day
        ST((Row + 1) Mod STArrayDim1, DSSubB).InflowM3 = _
            ST((Row + 1) Mod STArrayDim1, DSSubB).InflowM3 + DSOutflowOnRiverM3 *
RegimeRegionArray(Regime, Region).NextDayFractionalFlow
        If (WS(SubB).LakeID = 0) Then
            DepStorOnRiverM3 = WS(SubB).DepVolOnRiverM3  'Sets depressional storage equal
to the maximum
        Else 'this is a lake, reduce depressional storage by the outflow amount
            DepStorOnRiverM3 = DepStorOnRiverM3 - DSOutflowOnRiverM3
        End If
    Else
        DSOutflowOnRiverM3 = 0#
    End If

    If WS(SubB).ExtraOutflowIndex <> 0 Then
        rstExtraOutflowTable(WS(SubB).ExtraOutflowIndex).AddNew
        rstExtraOutflowTable(WS(SubB).ExtraOutflowIndex)![Date] = CurrentDate
        rstExtraOutflowTable(WS(SubB).ExtraOutflowIndex)![CMPD] = DSOutflowOnRiverM3
        rstExtraOutflowTable(WS(SubB).ExtraOutflowIndex).Update
    End If

    ' The following adds the Soil Capacity of this subwatershed times it's fraction of the
total non-depressional area
    ' When summed over entire watershed, SoilCapacityFraction is the weighted average of
the soil capacity of the watershed
    SoilCapacityFraction = SoilCapacityFraction + (SoilCapM / PotentialSoilCapM) *
WS(SubB).WSAreaNonDepFractionOfTotal
```

```
    'Prepare the ST Array for this subwatershed for the following day
    ST((Row + 1) Mod STArrayDim1, SubB).StartDepStorOnRiverM3 = DepStorOnRiverM3
    ST((Row + 1) Mod STArrayDim1, SubB).StartDepStorOffRiverM3 = DepStorOffRiverM3
    ST((Row + 1) Mod STArrayDim1, SubB).StartSoilCapM = SoilCapM

    If InfilM < 0# Or PercM < 0# Or SoilCapM > PotentialSoilCapM Then
        MsgBox ("Prohibited values being found -- break program and debug")
    End If

    'Increase the total water storage
    TotalWaterEndOfDayM3 = TotalWaterEndOfDayM3 + DepStorOnRiverM3 + DepStorOffRiverM3

    'Write out the trace record to the table if LongTrace button is set
    'used for debugging
    If LongTrace.Value = True Then
        With rst
           .AddNew
            ![Date] = CurrentDate
            ![SubBSeq] = SubB
            ![DSSubBSeq] = WS(SubB).DSSubBSeq
            ![Regime] = Regime
            ![PotentialInfilMPD] = PotentialInfilMPD
            ![PotentialPercMPD] = PotentialPercMPD
            ![ETCoef] = ETCoef
            ![UpperZoneM] = UpperZoneM
            ![PotentialSoilCapM] = PotentialSoilCapM
            ![PrecipM] = PrecipM
            ![InfilM] = InfilM
            ![PercM] = PercM
            ![StartSoilCapM] = ST(Row, SubB).StartSoilCapM
            ![EndSoilCapM] = SoilCapM
            ![SoilEvapM] = SoilEvapM
            ![EndSoilCapM] = SoilCapM
            ![InflowM3] = ST(Row, SubB).InflowM3
            ![StartDepStorOnRiverM3] = ST(Row, SubB).StartDepStorOnRiverM3
            ![StartDepStorOffRiverM3] = ST(Row, SubB).StartDepStorOffRiverM3
            ![PrecipOnDepOnRiverM3] = PrecipOnDepOnRiverM3
            ![PrecipOnDepOffRiverM3] = PrecipOnDepOffRiverM3
            ![RunoffToDepStorOnRiverM3] = RunoffToDepStorOnRiverM3
            ![RunoffToDepStorOffRiverM3] = RunoffToDepStorOffRiverM3
            ![DepEvapOnRiverM3] = DepEvapOnRiverM3
            ![DepEvapOffRiverM3] = DepEvapOffRiverM3
            ![EndDepStorOnRiverM3] = DepStorOnRiverM3
            ![EndDepStorOffRiverM3] = DepStorOffRiverM3
            ![DepAreaForEvapOnRiverM2] = DepAreaForEvapOnRiverM2
            ![DepAreaForEvapOffRiverM2] = DepAreaForEvapOffRiverM2
            ![DSOutflowOnRiverM3] = DSOutflowOnRiverM3
            ![OutflowOffRiverM3] = OutflowOffRiverM3
            .Update
        End With

    End If

Next SubB

'Write out the outlet flow if the LongTrace button is on
'for debugging
If LongTrace.Value = True Then
   With rst
       .AddNew
       ![Date] = CurrentDate
       ![SubBSeq] = 0
       ![InflowM3] = ST(Row, 0).InflowM3
       .Update
    End With
End If
```

```
'Write out the results to the SimulOutput table
With rstOutput
    .AddNew
    ![Date] = CurrentDate
    ![CMPD] = ST(Row, 0).InflowM3
     TotalOutletOutflowM3 = TotalOutletOutflowM3 + ST(Row, 0).InflowM3
    ![StorageVolUtil] = TotalWaterEndOfDayM3 / (TotalIntactDepStorOnRiverM3 +
TotalIntactDepStorOffRiverM3)
    ![SoilCapacityFraction] = SoilCapacityFraction
    ![PrecipM3] = TodaysPrecipM3
    ![LostToETM3] = TodaysETEvapM3
    ![LostToEvapM3] = TodaysDepEvapOnRiverM3 + TodaysDepEvapOffRiverM3
    ![LostToPercM3] = TodaysPercM3
    .Update
End With


Exit_CalcOneDayFlow:
    Exit Sub

Err_CalcOneDayFlow:
    MsgBox Err.Description
    Resume Exit_CalcOneDayFlow
End Sub

Sub CreateOutputTable()
    'This creates the output table for the model; copied from the SimulOutput table
    'The table is named according the wateshed, simulation, and restoration
    'level as set in the function GetOutputTableName

Dim dbs As DAO.Database
Dim qdf As DAO.QueryDef
Dim idx As DAO.Index
Dim tdf As DAO.TableDef
Dim fldIndex As DAO.Field
Dim strSQL As String
Dim OutputTableName As String


OutputTableName = GetOutputTableName(txtWaterShed.Value)

Set dbs = CurrentDb

DeleteTableWithoutWarning (OutputTableName)

strSQL = "SELECT * INTO " & OutputTableName & " FROM SimulOutput ORDER BY Date;"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute

Set tdf = dbs.TableDefs(OutputTableName)
Set idx = tdf.CreateIndex("PrimaryKey")
Set fldIndex = idx.CreateField("Date", dbDate)
idx.Fields.Append fldIndex
idx.Primary = True
tdf.Indexes.Append idx
dbs.TableDefs.Refresh


dbs.Close

End Sub

Function GetOutputTableName(PrefixName)
    'returns the output table name based on watershed, simulation, and restoration level
```

```
GetOutputTableName = "OUTPUT_" & PrefixName & "_" & txtSimulation.Value & "_" &
txtRestorationLevel.Value

End Function

Function GetInflowTableName(PrefixName)
    'returns the inflow table name based on watershed, simulation, and restoration level

GetInflowTableName = "INFLOW_" & PrefixName & "_" & txtSimulation.Value & "_" &
txtRestorationLevel.Value

End Function

Function GetOutflowTableName(PrefixName)
    'returns the output table name based on watershed, simulation, and restoration level

GetOutflowTableName = "OUTFLOW_" & PrefixName & "_" & txtSimulation.Value & "_" &
txtRestorationLevel.Value

End Function


Private Sub InitializeExternalVariables()
    'Initializes all external variables

ReDim ExtraInflowTable(25) As String
ReDim rstExtraInflowTable(25) As DAO.Recordset
ReDim ExtraOutflowTable(25) As String
ReDim rstExtraOutflowTable(25) As DAO.Recordset

NumPrecipGages = 0
NumRegions = 0
LongestLakeTableSize = 0
STArrayDim1 = 0
SimulDays = 0
NumSubBasins = 0
TotalIntactDepStorOnRiverM3 = 0#
TotalIntactDepStorOffRiverM3 = 0#
TotalNonDepAreaOnRiverM2 = 0#
TotalNonDepAreaOffRiverM2 = 0#
TotalOutletOutflowM3 = 0#
NumExtraInflowTables = 0
NumExtraOutflowTables = 0

ModelWshedTable = ""
ModelBaseCaseWShedTable = ""
ModelRiverTable = ""
ModelRegimeTable = ""

StartDate = 0
EndDate = 0

RestorationTransitionDate = #10/1/2002# 'the date at which the restoration
    'scenarios take effect

End Sub
Private Sub GetLakes()
    'retrieves the lakes table into the LakeTable() array

Dim dbs As DAO.Database
Dim rstLakeList As DAO.Recordset
Dim rstLakeTable As DAO.Recordset
Dim strSQL As String
Dim OutputTableName As String
Dim TableRow As Integer
Dim NumLakes As Long 'The number of lakes in the lakes table
```

```
LongestLakeTableSize = 0

Set dbs = CurrentDb

strSQL = "SELECT LakeID, Count(Lakes.LakeID) AS CountOfLakeID FROM Lakes Group BY LakeID
ORDER BY LakeID;"
Set rstLakeList = dbs.OpenRecordset(strSQL)
rstLakeList.MoveLast
NumLakes = rstLakeList.RecordCount
If NumLakes <> rstLakeList![LakeID] Then
    MsgBox ("Lakes out of sequence.  Start with 1 and number continuously.")
End If
rstLakeList.MoveFirst
If rstLakeList![LakeID] <> 1 Then
    MsgBox ("Lakes out of sequence.  First Lake must have ID of 1")
End If

Do Until rstLakeList.EOF
    LongestLakeTableSize = Max(LongestLakeTableSize, rstLakeList![CountOfLakeID])
    rstLakeList.MoveNext
Loop
rstLakeList.MoveFirst

ReDim LakeTable(NumLakes, LongestLakeTableSize + 1) 'dimensions it up to the biggest table
on the list

Do Until rstLakeList.EOF
    strSQL = "SELECT * FROM Lakes WHERE ((LakeID) = " & rstLakeList![LakeID] & ") ORDER BY
ID;"
    Debug.Print strSQL
    Set rstLakeTable = dbs.OpenRecordset(strSQL)
    rstLakeTable.MoveFirst
    TableRow = 1
    Do Until rstLakeTable.EOF
        LakeTable(rstLakeTable![LakeID], TableRow).VolumeM3 = rstLakeTable![VolumeM3]
        LakeTable(rstLakeTable![LakeID], TableRow).AreaM2 = rstLakeTable![AreaM2]
        LakeTable(rstLakeTable![LakeID], TableRow).FlowM3PD = rstLakeTable![FlowM3PD]
        TableRow = TableRow + 1
        rstLakeTable.MoveNext
    Loop
    rstLakeTable.Close
    rstLakeList.MoveNext
Loop

rstLakeList.Close
dbs.Close

End Sub

Function GetLakeTableRow(LakeID As Integer, VolumeM3 As Double)

Dim Row As Integer


For Row = 1 To LongestLakeTableSize + 1 'Cycles to the upper bound the Rows in LakeTable
    If LakeTable(LakeID, Row).VolumeM3 > VolumeM3 Then 'you've passed the row you wanted to
got to
        GetLakeTableRow = Max(1, Row - 1)
        Exit Function
    End If
    If Row > 1 And LakeTable(LakeID, Row).VolumeM3 = 0# Then 'you've gone past the end of
the table
        GetLakeTableRow = Row - 1
        Exit Function
    End If

Next
```

```
End Function
Private Sub CreateModel_Click()
    'Creates a base model for the simulation program based on five
    'exported tables from ArcView / HEC-GeoHMS

Dim dbs As DAO.Database
Dim rst As DAO.Recordset
Dim tdf As DAO.TableDef
Dim fld As DAO.Field
Dim qdf As DAO.QueryDef
Dim idx As DAO.Index
Dim fldIndex As DAO.Field
Dim strSQL As String

Dim ImpDepTable As String
Dim ImpPourPointWaterShedTable As String
Dim ImpPourPointLocationTable As String
Dim ImpWaterShedTable As String
Dim ImpRiverTable As String
Dim ImpPrefix As String

MsgBox ("Warning: This will wipe out any existing model for this watershed.  Cntrl-Break to
Cancel.")

Set dbs = CurrentDb

ImpPrefix = "IMP_" & txtWaterShed.Value & "_"

'Establishes the names of the 5 imported ArcView tables based on the model
'name and the restoration level
'The 1st table is the depression table, which also has affixed to its
'name the restoration level since the restoration levels affects the depression
'categories (DepType123)
ImpDepTable = ImpPrefix & "DEPTABLE" & "_" & txtRestorationLevel.Value
'The 2nd table is the the pour point watershed theme (which has had all
'depressions intact and drained trimed out of it.  It provides information
'regarding the total contributing area of each depression, excluding the depression
'itself and excluding other depressions intact or drained.
ImpPourPointWaterShedTable = ImpPrefix & "PPOINTWSHEDS"
'The 3rd table is from the pour points theme, it gives the pour point locations
'in terms of PolyID by WShid (watershed ID)
ImpPourPointLocationTable = ImpPrefix & "PPOINTLOCATIONS"
'The 4th table is the watershed table from HEC-GeoHMS.  It has the WShid
'(watershed ID) field, and the precipitation gage and region assignment for
'each watershed.
ImpWaterShedTable = ImpPrefix & "WSHEDS"
'The 5th table is the river table from HEC-GeoHMS, which gives provides
'sequencing information for the watersheds.
ImpRiverTable = ImpPrefix & "RIVER"

'The following 5 statements delete the temporary tables where the imported
'tables will be copied to.
DeleteTableWithoutWarning ("TEMP_IMP_DEPTABLE")
DeleteTableWithoutWarning ("TEMP_IMP_PPOINTWSHEDS")
DeleteTableWithoutWarning ("TEMP_IMP_PPOINTLOCATIONS")
DeleteTableWithoutWarning ("TEMP_IMP_WSHEDS")
DeleteTableWithoutWarning ("TEMP_IMP_RIVER")

'Puts the depression table for the model into a temporary table
strSQL = "SELECT * INTO TEMP_IMP_DEPTABLE FROM " & ImpDepTable & ";"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute
dbs.TableDefs.Refresh

'Puts the pour points watersheds table into a temporary table.
strSQL = "SELECT * INTO TEMP_IMP_PPOINTWSHEDS FROM " & ImpPourPointWaterShedTable & ";"
```

```
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute
dbs.TableDefs.Refresh

'Puts the pour point locations into a temporary table.
strSQL = "SELECT * INTO TEMP_IMP_PPOINTLOCATIONS FROM " & ImpPourPointLocationTable & ";"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute
dbs.TableDefs.Refresh

'Puts the model's imported watersheds into a temporary table.
strSQL = "SELECT * INTO TEMP_IMP_WSHEDS FROM " & ImpWaterShedTable & ";"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute
dbs.TableDefs.Refresh

'Puts the model's river table into a temporary table
strSQL = "SELECT * INTO TEMP_IMP_RIVER FROM " & ImpRiverTable & ";"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute
dbs.TableDefs.Refresh

Call SetModelTableNames 'sets all the table names for the model
    'e.g. those beginning with "MODEL_"

'Deletes the model's watershed table if it already exists
DeleteTableWithWarning (ModelWshedTable)

'The following deletes the model River table and Regime Table if is the
'original setup (e.g. restoration level "A") of the model.
If txtRestorationLevel.Value = "A" Then
    DeleteTableWithWarning (ModelRiverTable)
    DeleteTableWithWarning (ModelRegimeTable)
End If

'The following, until "dbs.TableDefs.Refresh," creates the model
'watershed table
Set tdf = dbs.CreateTableDef(ModelWshedTable)
Set fld = tdf.CreateField("SubBSeq", dbLong)
tdf.Fields.Append fld
Set fld = tdf.CreateField("WShid", dbLong)
tdf.Fields.Append fld
Set fld = tdf.CreateField("DSSubBSeq", dbLong)
tdf.Fields.Append fld
Set fld = tdf.CreateField("PrecipGage", dbText, 12)
tdf.Fields.Append fld
Set fld = tdf.CreateField("Region", dbLong)
tdf.Fields.Append fld
Set fld = tdf.CreateField("InitWaterM3", dbSingle)
tdf.Fields.Append fld
Set fld = tdf.CreateField("InputStreamTable", dbText, 12)
tdf.Fields.Append fld
Set fld = tdf.CreateField("LakeID", dbInteger)
tdf.Fields.Append fld
Set fld = tdf.CreateField("ExtraInflowID", dbText, 12)
tdf.Fields.Append fld
Set fld = tdf.CreateField("ExtraOutflowID", dbText, 12)
tdf.Fields.Append fld
Set fld = tdf.CreateField("WSAreaM2NonDep", dbSingle)
tdf.Fields.Append fld
Set fld = tdf.CreateField("DepVolM3", dbSingle)
tdf.Fields.Append fld
Set fld = tdf.CreateField("DepAreaM2", dbSingle)
tdf.Fields.Append fld
Set fld = tdf.CreateField("WSAreaNonDepOnRiverM2", dbSingle)
tdf.Fields.Append fld
Set fld = tdf.CreateField("DepVolOnRiverM3", dbSingle)
```

```
tdf.Fields.Append fld
Set fld = tdf.CreateField("DepAreaOnRiverM2", dbSingle)
tdf.Fields.Append fld
Set fld = tdf.CreateField("WSAreaNonDepOffRiverM2", dbSingle)
tdf.Fields.Append fld
Set fld = tdf.CreateField("DepVolOffRiverM3", dbSingle)
tdf.Fields.Append fld
Set fld = tdf.CreateField("DepAreaOffRiverM2", dbSingle)
tdf.Fields.Append fld
Set fld = tdf.CreateField("NumDeps", dbLong)
tdf.Fields.Append fld
Set fld = tdf.CreateField("NumDepsOnRiver", dbLong)
tdf.Fields.Append fld
Set fld = tdf.CreateField("NumDepsOffRiver", dbLong)
tdf.Fields.Append fld
Set idx = tdf.CreateIndex("PrimaryKey")
Set fldIndex = idx.CreateField("SubBSeq", dbLong)
idx.Fields.Append fldIndex
idx.Primary = True
tdf.Indexes.Append idx
dbs.TableDefs.Append tdf
dbs.TableDefs.Refresh

If txtRestorationLevel.Value = "A" Then ' Creates regime and river tables
        'for the original model setup e,g. restoration level "A"

    DeleteTableWithoutWarning ("RIVER") 'deletes the temporaty RIVER table

    ' Copies the model's Imported River table in RIVER
    strSQL = "SELECT * INTO RIVER FROM " & ImpRiverTable & ";"
    Set qdf = dbs.CreateQueryDef("", strSQL)
    qdf.Execute
    dbs.TableDefs.Refresh

    Set tdf = dbs.TableDefs("RIVER")

    'Adds several fields to the River table
    'The Layer field is the distance in river links from the outlet
    'The SubBSeq is the sequencing that will be used for the watershed
    'table.
    'DSSubBSeq is the downstream subbasin sequence number
    'DistFromOutM is not used at this time
    Set fld = tdf.CreateField("Layer", dbLong)
    tdf.Fields.Append fld
    Set fld = tdf.CreateField("SubBSeq", dbLong)
    tdf.Fields.Append fld
    Set fld = tdf.CreateField("DSSubBSeq", dbLong)
    tdf.Fields.Append fld
    Set fld = tdf.CreateField("DistFromOutM", dbDouble)
    tdf.Fields.Append fld
    dbs.TableDefs.Refresh

    'The following function takes the to and from nodes in the
    'RIVER table and assigns values to Layer, SubBSeq, and DSSubSeq
    'Such that all the segements are sequenced properly, and so that
    'and watershed with a certain SubBSeq number is guaranteed not to
    'be downstream of any watershed with a higher SubBSeq.  That is, the
    'lower SubBSeq numbers correspond to the most upstream watersheds
    Call SequenceRiver_Click

    ' Takes the date from the RIVER table and copies it into the
    'model's river table.
    strSQL = "SELECT * INTO " & ModelRiverTable & " FROM RIVER;"
    Set qdf = dbs.CreateQueryDef("", strSQL)
    qdf.Execute
    dbs.TableDefs.Refresh
```

```
    'Creates a default regime table for the model using the table
    'RegimesTemplate as a source
    strSQL = "SELECT * INTO " & ModelRegimeTable & " FROM RegimesTemplate;"
    Set qdf = dbs.CreateQueryDef("", strSQL)
    qdf.Execute
    dbs.TableDefs.Refresh

End If

'Creates the watersheds for the watersheds table using the sequencing
'information previously calculated which is now in the model river table
'This query fills the SubBSeq and DSSubBSeq fields only
strSQL = "INSERT INTO " & ModelWshedTable & " ( SubBSeq, DSSubBSeq ) " & _
        "SELECT SubBSeq, DSSubBSeq FROM " & ModelRiverTable & ";"
Set qdf = dbs.CreateQueryDef("", strSQL)
Debug.Print strSQL
dbs.QueryDefs.Refresh
qdf.Execute

'Updates the WShid of the model watershed table based on the SubBSeq / Wshid
'relationship which appears in the model river table.
strSQL = "UPDATE " & ModelWshedTable & " INNER JOIN " & ModelRiverTable & _
        " ON " & ModelWshedTable & ".SubBSeq = " & ModelRiverTable & ".SubBSeq SET " & _
        ModelWshedTable & ".WShid = " & ModelRiverTable & ".[WShid];"
Set qdf = dbs.CreateQueryDef("", strSQL)
dbs.QueryDefs.Refresh
qdf.Execute

'Deletes the temporary watershed table
DeleteTableWithoutWarning ("TEMP_MODEL_WSHEDS")
'Creates a temporary watershed table using the information currently in
'the model's watershed table: the SubBSeq, DSSubBSeq, and WShid, as well
'as numerous other fields with no data
strSQL = "SELECT * INTO TEMP_MODEL_WSHEDS FROM " & ModelWshedTable & ";"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute
dbs.TableDefs.Refresh

' Finds the areas of Non Type 1 depressions in the watershed
DeleteQueryWithoutWarning ("TempQNonType1Area")
strSQL = "SELECT TEMP_MODEL_WSHEDS.SubBSeq, Sum(TEMP_IMP_DEPTABLE.AREA_M2) AS SumOfAREA_M2
" & _
        "FROM (TEMP_MODEL_WSHEDS INNER JOIN TEMP_IMP_PPOINTLOCATIONS ON
TEMP_MODEL_WSHEDS.WShid = TEMP_IMP_PPOINTLOCATIONS.WSHID) INNER JOIN TEMP_IMP_DEPTABLE ON
TEMP_IMP_PPOINTLOCATIONS.POLYID = TEMP_IMP_DEPTABLE.POLYID " & _
        "GROUP BY TEMP_MODEL_WSHEDS.SubBSeq, TEMP_IMP_DEPTABLE.DEPTYPE123 " & _
        "HAVING (((TEMP_IMP_DEPTABLE.DEPTYPE123)<>1));"
Set qdf = dbs.CreateQueryDef("TempQNonType1Area", strSQL)
dbs.QueryDefs.Refresh

' Finds Soil (contributing) each watershed
DeleteQueryWithoutWarning ("TempQSoilAreas")
strSQL = "SELECT TEMP_MODEL_WSHEDS.SubBSeq, Sum(TEMP_IMP_PPOINTWSHEDS.AREA_M2) AS
SoilAreaM2 " & _
        "FROM (TEMP_IMP_PPOINTWSHEDS INNER JOIN TEMP_IMP_PPOINTLOCATIONS ON
TEMP_IMP_PPOINTWSHEDS.POLYID = TEMP_IMP_PPOINTLOCATIONS.POLYID) INNER JOIN
TEMP_MODEL_WSHEDS ON TEMP_IMP_PPOINTLOCATIONS.WSHID = TEMP_MODEL_WSHEDS.WShid " & _
        "GROUP BY TEMP_MODEL_WSHEDS.SubBSeq;"
Set qdf = dbs.CreateQueryDef("TempQSoilAreas", strSQL)
dbs.QueryDefs.Refresh

' Unions the Contributing areas (soils) with the Non Type 1 depressions to create total
contributing area
DeleteQueryWithoutWarning ("TempQUnionNonType1Area")
strSQL = "SELECT ALL * FROM TempQSoilAreas UNION ALL SELECT ALL * FROM TempQNonType1Area;"
Set qdf = dbs.CreateQueryDef("TempQUnionNonType1Area", strSQL)
dbs.QueryDefs.Refresh
```

```
' Writes out the total contributing area to a Sub Basin
DeleteTableWithoutWarning ("TEMP_TOTALNONTYPE1AREA")
strSQL = "SELECT SubBSeq, Sum(SoilAreaM2) AS SumOfSoilAreaM2 INTO TEMP_TOTALNONTYPE1AREA "
& _
        "FROM TempQUnionNonType1Area " & _
        "GROUP BY SubBSeq " & _
        "ORDER BY SubBSeq;"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute
dbs.TableDefs.Refresh

' Creates a query that determines the Soil contributing areas, classified by INTERRIV =
True or False
DeleteQueryWithoutWarning ("TempQSoilAreasByInterRiv")
strSQL = "SELECT TEMP_MODEL_WSHEDS.SubBSeq, Sum(TEMP_IMP_PPOINTWSHEDS.AREA_M2) AS
SoilAreaM2, TEMP_IMP_DEPTABLE.INTERRIV " & _
        "FROM ((TEMP_IMP_PPOINTWSHEDS INNER JOIN TEMP_IMP_PPOINTLOCATIONS ON
TEMP_IMP_PPOINTWSHEDS.POLYID = TEMP_IMP_PPOINTLOCATIONS.POLYID) INNER JOIN
TEMP_MODEL_WSHEDS ON TEMP_IMP_PPOINTLOCATIONS.WSHID = TEMP_MODEL_WSHEDS.WShid) INNER JOIN
TEMP_IMP_DEPTABLE ON TEMP_IMP_PPOINTWSHEDS.POLYID = TEMP_IMP_DEPTABLE.POLYID " & _
        "GROUP BY TEMP_MODEL_WSHEDS.SubBSeq, TEMP_IMP_DEPTABLE.INTERRIV;"
Set qdf = dbs.CreateQueryDef("TempQSoilAreasByInterRiv", strSQL)
dbs.QueryDefs.Refresh

' Creates a query that determines the non-Type 1 depressional contributing area, classified
by River intersection or not
DeleteQueryWithoutWarning ("TempQNonType1AreaByInterRiv")
strSQL = "SELECT TEMP_MODEL_WSHEDS.SubBSeq, Sum(TEMP_IMP_DEPTABLE.AREA_M2) AS SumOfAREA_M2,
TEMP_IMP_DEPTABLE.INTERRIV " & _
        "FROM (TEMP_MODEL_WSHEDS INNER JOIN TEMP_IMP_PPOINTLOCATIONS ON
TEMP_MODEL_WSHEDS.WShid = TEMP_IMP_PPOINTLOCATIONS.WSHID) INNER JOIN TEMP_IMP_DEPTABLE ON
TEMP_IMP_PPOINTLOCATIONS.POLYID = TEMP_IMP_DEPTABLE.POLYID " & _
        "GROUP BY TEMP_MODEL_WSHEDS.SubBSeq, TEMP_IMP_DEPTABLE.INTERRIV,
TEMP_IMP_DEPTABLE.DEPTYPE123 " & _
        "HAVING (((TEMP_IMP_DEPTABLE.DEPTYPE123)<>1));"
Set qdf = dbs.CreateQueryDef("TempQNonType1AreaByInterRiv", strSQL)
dbs.QueryDefs.Refresh

'Creates a query that determines the outlet contributing area, this together with previous
2
'queries are unioned in the query following this one
DeleteQueryWithoutWarning ("TempQOutletByInterRiv")
strSQL = "SELECT TEMP_MODEL_WSHEDS.SubBSeq, TEMP_IMP_PPOINTWSHEDS.AREA_M2, True AS INTERRIV
" & _
        "FROM TEMP_MODEL_WSHEDS, TEMP_IMP_PPOINTWSHEDS " & _
        "WHERE (((TEMP_MODEL_WSHEDS.DSSubBSeq)=0) AND ((TEMP_IMP_PPOINTWSHEDS.POLYID)=0));"
Set qdf = dbs.CreateQueryDef("TempQOutletByInterRiv", strSQL)
dbs.QueryDefs.Refresh

' Creates a query that Unions the Contributing areas (soils) with the Non Type 1
depressions to create total contributing area
' They are classified by River intersection (INTERRIV = True Or False)
DeleteQueryWithoutWarning ("TempQUnionNonType1AreaByInterRiv")
strSQL = "SELECT ALL * FROM TempQSoilAreasByInterRiv UNION ALL SELECT ALL * FROM
TempQNonType1AreaByInterRiv UNION ALL SELECT ALL * FROM TempQOutletByInterRiv;"
Set qdf = dbs.CreateQueryDef("TempQUnionNonType1AreaByInterRiv", strSQL)
dbs.QueryDefs.Refresh

'Deletes a temporary table before creating a new version
DeleteTableWithoutWarning ("TEMP_TOTALNONTYPE1AREA_INTERRIV")
'Creates a table that gives total contributing areas to each watershed classified by River
Intersection of Depressions
strSQL = "SELECT SubBSeq, Sum(SoilAreaM2) AS SumOfSoilAreaM2, INTERRIV INTO
TEMP_TOTALNONTYPE1AREA_INTERRIV " & _
        "FROM TempQUnionNonType1AreaByInterRiv " & _
        "GROUP BY SubBSeq, INTERRIV " & _
```

```
        "ORDER BY SubBSeq;"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute

'Updates the temporary watershed table with the non-type 1 areas
strSQL = "UPDATE TEMP_TOTALNONTYPE1AREA INNER JOIN TEMP_MODEL_WSHEDS ON
TEMP_TOTALNONTYPE1AREA.SubBSeq = TEMP_MODEL_WSHEDS.SubBSeq SET
TEMP_MODEL_WSHEDS.WSAreaM2NonDep = [SumOfSoilAreaM2];"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute

'Deletes a temporary table before creating a new version
DeleteTableWithoutWarning ("TEMP_INTACTAREASVOLUMES")
'Creates a table which has the volumes and areas of all intact depressions
strSQL = "SELECT TEMP_MODEL_WSHEDS.SubBSeq, Sum(TEMP_IMP_DEPTABLE.AREA_M2) AS SumOfAREA_M2,
Sum(TEMP_IMP_DEPTABLE.VOLM3) AS SumOfVOLM3, Count(TEMP_IMP_DEPTABLE.POLYID) AS
CountOfPOLYID INTO TEMP_INTACTAREASVOLUMES " & _
        "FROM (TEMP_IMP_DEPTABLE INNER JOIN TEMP_IMP_PPOINTLOCATIONS ON
TEMP_IMP_DEPTABLE.POLYID = TEMP_IMP_PPOINTLOCATIONS.POLYID) INNER JOIN TEMP_MODEL_WSHEDS ON
TEMP_IMP_PPOINTLOCATIONS.WSHID = TEMP_MODEL_WSHEDS.WShid " & _
        "GROUP BY TEMP_MODEL_WSHEDS.SubBSeq, TEMP_IMP_DEPTABLE.DEPTYPE123 " & _
        "HAVING (((TEMP_IMP_DEPTABLE.DEPTYPE123)=1));"
Set qdf = dbs.CreateQueryDef("", strSQL)
Debug.Print strSQL
qdf.Execute
dbs.TableDefs.Refresh

'Updates the temporary watersheds table with the depressional areas and
'volumes contained in TEMP_INTACTAREASVOLUMES
strSQL = "UPDATE TEMP_INTACTAREASVOLUMES INNER JOIN TEMP_MODEL_WSHEDS ON
TEMP_INTACTAREASVOLUMES.SubBSeq = TEMP_MODEL_WSHEDS.SubBSeq SET TEMP_MODEL_WSHEDS.DepVolM3
= [SumOfVOLM3], TEMP_MODEL_WSHEDS.DepAreaM2 = [SumOfAREA_M2], TEMP_MODEL_WSHEDS.NumDeps =
[CountOfPOLYID];"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute

' Creates table with depressional volumes and areas on the river
DeleteTableWithoutWarning ("TEMP_INTACTAREASVOLUMES_ONRIVER")
strSQL = "SELECT TEMP_MODEL_WSHEDS.SubBSeq, Sum(TEMP_IMP_DEPTABLE.AREA_M2) AS SumOfAREA_M2,
Sum(TEMP_IMP_DEPTABLE.VOLM3) AS SumOfVOLM3, Count(TEMP_IMP_DEPTABLE.POLYID) AS
CountOfPOLYID INTO TEMP_INTACTAREASVOLUMES_ONRIVER " & _
        "FROM (TEMP_IMP_DEPTABLE INNER JOIN TEMP_IMP_PPOINTLOCATIONS ON
TEMP_IMP_DEPTABLE.POLYID = TEMP_IMP_PPOINTLOCATIONS.POLYID) INNER JOIN TEMP_MODEL_WSHEDS ON
TEMP_IMP_PPOINTLOCATIONS.WSHID = TEMP_MODEL_WSHEDS.WShid " & _
        "GROUP BY TEMP_MODEL_WSHEDS.SubBSeq, TEMP_IMP_DEPTABLE.DEPTYPE123,
TEMP_IMP_DEPTABLE.INTERRIV " & _
        "HAVING (((TEMP_IMP_DEPTABLE.DEPTYPE123)=1) AND
((TEMP_IMP_DEPTABLE.INTERRIV)=True));"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute

' Creates table with depressional volumes and areas off the river
DeleteTableWithoutWarning ("TEMP_INTACTAREASVOLUMES_OFFRIVER")
strSQL = "SELECT TEMP_MODEL_WSHEDS.SubBSeq, Sum(TEMP_IMP_DEPTABLE.AREA_M2) AS SumOfAREA_M2,
Sum(TEMP_IMP_DEPTABLE.VOLM3) AS SumOfVOLM3, Count(TEMP_IMP_DEPTABLE.POLYID) AS
CountOfPOLYID INTO TEMP_INTACTAREASVOLUMES_OFFRIVER " & _
        "FROM (TEMP_IMP_DEPTABLE INNER JOIN TEMP_IMP_PPOINTLOCATIONS ON
TEMP_IMP_DEPTABLE.POLYID = TEMP_IMP_PPOINTLOCATIONS.POLYID) INNER JOIN TEMP_MODEL_WSHEDS ON
TEMP_IMP_PPOINTLOCATIONS.WSHID = TEMP_MODEL_WSHEDS.WShid " & _
        "GROUP BY TEMP_MODEL_WSHEDS.SubBSeq, TEMP_IMP_DEPTABLE.DEPTYPE123,
TEMP_IMP_DEPTABLE.INTERRIV " & _
        "HAVING (((TEMP_IMP_DEPTABLE.DEPTYPE123)=1) AND
((TEMP_IMP_DEPTABLE.INTERRIV)=False));"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute
```

```
' Updates the temporary Watershed table with the contributing areas for depressions on the
River
strSQL = "UPDATE TEMP_TOTALNONTYPE1AREA_INTERRIV INNER JOIN TEMP_MODEL_WSHEDS ON
TEMP_TOTALNONTYPE1AREA_INTERRIV.SubBSeq = TEMP_MODEL_WSHEDS.SubBSeq SET
TEMP_MODEL_WSHEDS.WSAreaNonDepOnRiverM2 = [SumOfSoilAreaM2] " & _
        "WHERE (((TEMP_TOTALNONTYPE1AREA_INTERRIV.INTERRIV)=True));"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute

' Updates the temporary Watershed table with the contributing areas for depressions off the
River
strSQL = "UPDATE TEMP_TOTALNONTYPE1AREA_INTERRIV INNER JOIN TEMP_MODEL_WSHEDS ON
TEMP_TOTALNONTYPE1AREA_INTERRIV.SubBSeq = TEMP_MODEL_WSHEDS.SubBSeq SET
TEMP_MODEL_WSHEDS.WSAreaNonDepOffRiverM2 = [SumOfSoilAreaM2] " & _
        "WHERE (((TEMP_TOTALNONTYPE1AREA_INTERRIV.INTERRIV)=False));"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute

'Update the temporary watershed table with the On River depressional volumes and areas
strSQL = "UPDATE TEMP_INTACTAREASVOLUMES_ONRIVER INNER JOIN TEMP_MODEL_WSHEDS ON
TEMP_INTACTAREASVOLUMES_ONRIVER.SubBSeq = TEMP_MODEL_WSHEDS.SubBSeq SET
TEMP_MODEL_WSHEDS.DepVolOnRiverM3 = [SumOfVOLM3], TEMP_MODEL_WSHEDS.DepAreaOnRiverM2 =
[SumOfAREA_M2], TEMP_MODEL_WSHEDS.NumDepsOnRiver = [CountOfPolyID];"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute

'Update the temporary watershed table with the Off River depressional volumes and areas
strSQL = "UPDATE TEMP_MODEL_WSHEDS INNER JOIN TEMP_INTACTAREASVOLUMES_OFFRIVER ON
TEMP_MODEL_WSHEDS.SubBSeq = TEMP_INTACTAREASVOLUMES_OFFRIVER.SubBSeq SET
TEMP_MODEL_WSHEDS.DepVolOffRiverM3 = [SumOfVOLM3], TEMP_MODEL_WSHEDS.DepAreaOffRiverM2 =
[SumOfAREA_M2], TEMP_MODEL_WSHEDS.NumDepsOffRiver = [CountOfPolyID];"
Set qdf = dbs.CreateQueryDef("", strSQL)
Debug.Print strSQL
qdf.Execute

'updates the Gage Names and Regions in the Temporary WSHEDS table
strSQL = "UPDATE TEMP_IMP_WSHEDS INNER JOIN TEMP_MODEL_WSHEDS ON TEMP_IMP_WSHEDS.WSHID =
TEMP_MODEL_WSHEDS.WShid SET TEMP_MODEL_WSHEDS.PrecipGage = [TEMP_IMP_WSHEDS].[PrecipGage],
TEMP_MODEL_WSHEDS.Region = [TEMP_IMP_WSHEDS].[Region];"
Set qdf = dbs.CreateQueryDef("", strSQL)
Debug.Print strSQL
qdf.Execute

'Updates the WSheds Table for this model
strSQL = "UPDATE TEMP_MODEL_WSHEDS INNER JOIN " & ModelWshedTable & " AS MW ON
TEMP_MODEL_WSHEDS.SubBSeq = MW.SubBSeq SET MW.PrecipGage =
[TEMP_MODEL_WSHEDS].[PrecipGage], MW.WSAreaM2NonDep = [TEMP_MODEL_WSHEDS].[WSAreaM2NonDep],
MW.DepVolM3 = [TEMP_MODEL_WSHEDS].[DepVolM3], MW.DepAreaM2 =
[TEMP_MODEL_WSHEDS].[DepAreaM2], MW.WSAreaNonDepOnRiverM2 =
[TEMP_MODEL_WSHEDS].[WSAreaNonDepOnRiverM2], MW.DepVolOnRiverM3 =
[TEMP_MODEL_WSHEDS].[DepVolOnRiverM3], MW.DepAreaOnRiverM2 =
[TEMP_MODEL_WSHEDS].[DepAreaOnRiverM2], MW.WSAreaNonDepOffRiverM2 =
[TEMP_MODEL_WSHEDS].[WSAreaNonDepOffRiverM2], MW.DepVolOffRiverM3 =
[TEMP_MODEL_WSHEDS].[DepVolOffRiverM3], MW.DepAreaOffRiverM2 =
[TEMP_MODEL_WSHEDS].[DepAreaOffRiverM2], MW.NumDeps = [TEMP_MODEL_WSHEDS].[NumDeps],
MW.NumDepsOnRiver = [TEMP_MODEL_WSHEDS].[NumDepsOnRiver], MW.NumDepsOffRiver =
[TEMP_MODEL_WSHEDS].[NumDepsOffRiver];"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute

'Updates the Wsheds table regions for this model
strSQL = "UPDATE TEMP_MODEL_WSHEDS INNER JOIN " & ModelWshedTable & " AS MW ON
TEMP_MODEL_WSHEDS.SubBSeq = MW.SubBSeq SET MW.Region = [TEMP_MODEL_WSHEDS].[Region];"
Set qdf = dbs.CreateQueryDef("", strSQL)
Debug.Print strSQL
qdf.Execute
```

```
'Opens the model watershed table
strSQL = "SELECT * FROM " & ModelWshedTable & ";"
Set rst = dbs.OpenRecordset(strSQL)
rst.MoveFirst

With rst 'searches for null values in various fields of the watershed
        'table and replaces them with zeros.
    Do Until .EOF
        .Edit
        If IsNull(![DepVolM3]) Then
            ![DepVolM3] = 0#
        End If
        If IsNull(![DepAreaM2]) Then
            ![DepAreaM2] = 0#
        End If
        If IsNull(![NumDeps]) Then
            ![NumDeps] = 0
        End If
        If IsNull(![WSAreaM2NonDep]) Then
            ![WSAreaM2NonDep] = 0#
        End If
        If IsNull(![DepVolOnRiverM3]) Then
            ![DepVolOnRiverM3] = 0#
        End If
        If IsNull(![DepAreaOnRiverM2]) Then
            ![DepAreaOnRiverM2] = 0#
        End If
        If IsNull(![NumDepsOnRiver]) Then
            ![NumDepsOnRiver] = 0
        End If
        If IsNull(![DepVolOffRiverM3]) Then
            ![DepVolOffRiverM3] = 0#
        End If
        If IsNull(![DepAreaOffRiverM2]) Then
            ![DepAreaOffRiverM2] = 0#
        End If
        If IsNull(![NumDepsOffRiver]) Then
            ![NumDepsOffRiver] = 0
        End If
        If IsNull(![WSAreaNonDepOnRiverM2]) Then
            ![WSAreaNonDepOnRiverM2] = 0#
        End If
        If IsNull(![WSAreaNonDepOffRiverM2]) Then
            ![WSAreaNonDepOffRiverM2] = 0#
        End If
        If (IsNull(![Region])) Then 'sets Region = 1 if it was not set
                'previously
            ![Region] = 1
        End If
        ![InitWaterM3] = 0#
        .Update
        .MoveNext
    Loop
End With

rst.Close
dbs.Close

MsgBox ("Done Creating Model From ArcView Tables")

End Sub

Sub DeleteTableWithWarning(TableName As String)
    'Deletes a table without providing a warning

If DoesObjectExist("Tables", TableName) <> "" Then
    MsgBox ("Warning: The table " & TableName & " Will be Deleted.  Cntrl-Break to Cancel")
```

```
    DoCmd.DeleteObject acTable, TableName
End If

End Sub

Sub DeleteQueryWithoutWarning(QueryName As String)
    'Deletes a query without providing a warning

If DoesObjectExist("Queries", QueryName) <> "" Then
    DoCmd.DeleteObject acQuery, QueryName
End If

End Sub
Sub DeleteTableWithoutWarning(TableName As String)
    'Deletes a table after providing a warning

If DoesObjectExist("Tables", TableName) <> "" Then
    DoCmd.DeleteObject acTable, TableName
End If

End Sub

Private Sub SequenceRiver_Click()
    'sequences the RIVER table from upstream to downstream
    'and provides the subbasin sequencing numbers (SubBSeq)
    'that insure that no subbasin downstream of another subbasin
    'will have a lower SubBSeq

Dim dbs As DAO.Database
Dim rst As DAO.Recordset
Dim rstUpdate As DAO.Recordset
Dim qdf As DAO.QueryDef
Dim I As Long, Count As Long
Dim strSQL As String, strSQLUpdate As String
Dim OutletNode As Long
Dim CurrentLayer As Long, NumberOfLayers As Long, OldLayer As Long

StatusBox.Value = "Sequencing the RIVER table"
Me.Refresh

Set dbs = CurrentDb

'Clear the River.Layer, River.SubBSeq, and River.DSSubBSeq fields
strSQL = "UPDATE River SET River.Layer = Null, River.SubBSeq = Null, River.DSSubBSeq =
Null;"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute

'A query to find all River segments whose TO_NODE have no
'corresponding FROM_NODE in another River segement
'the record selected should be the outlet
strSQL = "SELECT River.[TO_NODE], River_1.[FROM_NODE] " & _
    "FROM River LEFT JOIN River AS River_1 ON River.[TO_NODE] = River_1.[FROM_NODE] " & _
    "WHERE (((River_1.[FROM_NODE]) Is Null));"

Set rst = dbs.OpenRecordset(strSQL)

rst.MoveLast
Count = rst.RecordCount 'hown many records were returned
OutletNode = rst![TO_NODE] 'This is the node number of the outlet
rst.Close

If Count <> 1 Then 'there should have been only one outlet returned
    MsgBox ("The outlet count was not 1 but: " & Count)
    Exit Sub
End If
```

```
'A query which finds the river segment whose TO_NODE is the outlet
'node; that is, this query returns the most downstream river segment
strSQL = "SELECT * FROM River WHERE (((TO_NODE)=" & OutletNode & "));"
Set rst = dbs.OpenRecordset(strSQL)

'Sets [Layer] equal to 1 for the the most downstream river segement
rst.Edit
rst![Layer] = 1
rst.Update
rst.Close

CurrentLayer = 1 'sets the intial layer to be 1'; this will cause the
    'following Do loop to select the outlet subbasin in it's first loop

Do 'This creates the Layer values, where Layer is the number of
    'subbasins away from the outlet; the most downstream subbasin
    'was already set to have a Layer value of 1
    StatusBox.Value = "Sequencing the RIVER table -- Layer = " & CurrentLayer
    Me.Refresh

    'extracts all subbasins in the current Layer
    strSQL = "SELECT River_1.* FROM River AS River_1 " & _
        "INNER JOIN River ON River_1.TO_NODE = River.FROM_NODE " & _
        "WHERE (((River.Layer)=" & CurrentLayer & "));"

    Set rst = dbs.OpenRecordset(strSQL)

    'If there are no further records quit the Do loop
    If rst.EOF Then
        rst.Close
        Exit Do
    End If

    rst.MoveFirst 'moves to the first subbasin extracted
    Do Until rst.EOF 'extracts all subbasins upstream of the current subbasin
        strSQLUpdate = "SELECT * FROM RIVER WHERE (((RIVID)=" & rst![RIVID] & "));"
        Set rstUpdate = dbs.OpenRecordset(strSQLUpdate)
        rstUpdate.Edit
        rstUpdate![Layer] = CurrentLayer + 1 'sets their layer value one higher
        rstUpdate.Update
        rstUpdate.Close
        rst.MoveNext
    Loop

    rst.Close
    CurrentLayer = CurrentLayer + 1 'increments the current layer
Loop

'Find the maximum Layer number
strSQL = "SELECT Max(River.Layer) AS MaxOfLayer FROM River;"
Set rst = dbs.OpenRecordset(strSQL)
rst.MoveFirst
NumberOfLayers = rst![MaxOfLayer]
rst.Close

'Assign the SubBasin Sequence numbers SubBSeq
'extracts the RIVER records sorted in reverse (DESC) order
'of the layer number so that the most upstream subbasins
'appear first
strSQL = "SELECT * FROM River ORDER BY Layer DESC, RIVID;"
Set rst = dbs.OpenRecordset(strSQL)
rst.MoveFirst
I = 1
Do Until rst.EOF
    rst.Edit
    rst![SubBSeq] = I
    rst.Update
```

```
    rst.MoveNext
    I = I + 1
Loop
rst.Close

'Assign the DownStream SubBasin Sequence number DSSubBSeq
'based on the node numbers
strSQL = "UPDATE River INNER JOIN River AS River_1 " & _
        "ON River.FROM_NODE = River_1.TO_NODE SET River_1.DSSubBSeq = [River].[SubBSeq];"
Set qdf = dbs.CreateQueryDef("", strSQL)
qdf.Execute

' Set the most downstream subbasin to have DSSubBSeq = 0
strSQL = "SELECT * FROM RIVER WHERE ([DSSubBSeq] Is Null)"
Set rst = dbs.OpenRecordset(strSQL)
rst.MoveFirst
rst.Edit
rst![DSSubBSeq] = 0
rst.Update
rst.Close

dbs.Close

End Sub

Sub SetModelTableNames()
    'Establishes the name of all input tables for the model
    'based on the watershed name, restoration level, and simulation to
    'be run.

Dim ModelPrefix As String

ModelPrefix = "MODEL_" & txtWaterShed.Value & "_"

ModelWshedTable = ModelPrefix & txtRestorationLevel.Value & "_" & "WSHEDS"
ModelBaseCaseWShedTable = ModelPrefix & "A_" & "WSHEDS"
ModelRiverTable = ModelPrefix & "RIVER"
ModelRegimeTable = ModelPrefix & "REGIMES"

End Sub

Private Function IsField(td As TableDef, ByVal FieldName As String) _
        As Boolean

' Returns TRUE if a field exists in the table with the same name as
'    specified in FieldName.
' Returns FALSE otherwise.

   Dim F As Field
   Err.Clear
   On Error Resume Next
   Set F = td(FieldName)
   If Err.Number = 13 Then
        IsField = True
    Else
        IsField = False
    End If
   Err.Clear
End Function
```